City Research Online

City, University of London Institutional Repository

Citation: Li, C.W. (1990). On-line distributed hierarchical control and optimisation of large scale systems. (Unpublished Doctoral thesis, City University London)

This is the accepted version of the paper.

This version of the publication may differ from the final published version.

Permanent repository link: https://openaccess.city.ac.uk/id/eprint/7528/

Link to published version:

Copyright: City Research Online aims to make research outputs of City, University of London available to a wider audience. Copyright and Moral Rights remain with the author(s) and/or copyright holders. URLs from City Research Online may be freely distributed and linked to.

Reuse: Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

City Research Online: http://openaccess.city.ac.uk/ publications@city.ac.uk

On-Line Distributed Hierarchical Control and Optimisation of Large Scale Systems

By

Chung Wai LI

Thesis submitted for the award of the degree
of
Doctor of Philosophy

City University
London

Control Engineering Centre

March, 1990

BEST COPY

AVAILABLE

TEXT IN ORIGINAL IS CLOSE TO THE EDGE OF THE PAGE

CONTENTS

ACKNOWLEDGEMENTS

I would like to express my greatest appreciation to my project supervisor, Professor P.D. Roberts for his invaluable advice, guidance and support throughout the course of my research.

I am also indebted to Mr. D.S. Wadhwani, Dr. I.A. Stevenson and Dr. J.E. Ellis for their help and valuable discussions in the Computer Control Laboratory.

Also, I would like to thank all members of staff and friends in the Control Engineering Centre who have contributed to this research.

Finally, I am grateful for the financial support from the Science and Engineering Research Council.

DECLARATION

The author grants powers of discretion to the University Librarian to allow this thesis to be copied in whole or part without further reference to the author. This permission covers only single copies made for study purposes, subject to normal conditions of acknowledgement.

ABSTRACT

This research is concerned with the application of closed-loop coordination techniques for on-line steady state optimisation and control of large scale systems using a micro-computer based system. A two-level hierarchical computer structure consisting of a coordinator at the supremal (upper) level and two local decision units at the infimal (lower) level had been established. Parallel computation were performed at the local decision unit level once the coordination parameters had been received from the supremal level. A steady state system model consisting of two interconnected subprocesses, simulated by an analogue computer, was used to investigate the coordination methods for closed-loop hierarchical control. First-order time constants were introduced to the interaction inputs and the controls within the simulated subprocesses.

Investigations had been carried out to study closed-loop control using the Interaction Prediction and Interaction Balance coordination method. Special attention was given to the study of the problems associated with synchronisation of the local decision units for closed-loop control. Stability aspects of both coordination methods when subjected to disturbances in the controls and interconnections had been investigated. Problems relating to system transient and local decision asynchronisation, as well as their effects on system stability and convergence of the two tasks, namely the local decision task and the coordinator task had also been investigated. Methods for dealing with these problems had been suggested. The sub-optimality, convergency and robustness properties of each coordination method had been discussed. This research has demonstrated that the Interaction Prediction coordination methods are best suited for on-line distributed optimising control of large scale interconnected systems. Using the local feedback scheme, complete decentralisation at the local decision level operated asynchronously can be achieved with the Interaction Prediction coordination method.

SYMBOLS AND/OR ABBREVIATIONS

c	Control input
COP	Coordinator Optimisation Problem
DHCS	Distributed Hierarchical Computer System
F	Input/output mapping (model)
F_*	Input/output mapping (real system)
G	Constraint set
H	Interconnection matrix
IBM	Interaction Balance Method
IBMGF	Interaction Balance Method with Global Feedback
IBMLF	Interaction Balance Method with Local Feedback
I-MIC	Industrial MIcro-Controller
IPM	Interaction Prediction Method
IPMGF	Interaction Prediction Method with Global Feedback
IPMLF	Interaction Prediction Method with Local Feedback
K,ε	System gain matrix
L,λ	Lagrangian
LDU	Local Decision Unit
LOP	Local Optimisation Problem
Q	Performance index (model)
Q_*	Performance index (real system)
ΔQ_*	Suboptimility
s	Shift vector
u	Interaction input (model)
u_*	Interaction input (real system)
v	Coordination variables for IPM
y	Interaction output (model)
y_*	Interaction output (real system)
z	Disturbances

CHAPTER 1 INTRODUCTION

There has been a growing attention paid to the subject area of large scale systems. This comes quite naturally as many real-life problems of socio-economic, environmental and technological nature are highly complex, large in dimension and stochastic in nature. Despite its generality and usefulness, the multivariable system approach is severely limited when applied to problems of high dimension and complex interconnecting structure. For this reason it is frequently advantageous to view high order systems as being composed of several lower order subsystems which when interconnected in an appropriate fashion, yield the original composite or interconnected system.

Many viewpoints have been put forward to define and quantify a system as 'large scale'. One viewpoint considers 'large scale systems' as those whose dimensions are so large that conventional techniques of modelling, analysis, control and optimisation are extremely difficult or impossible to give a reasonable solution. Another viewpoint suggests that if a system can be decomposed into a number of interconnected subsystems for computational and practical purposes, then such systems are termed as large scale systems. The latter viewpoint is adopted to define large scale systems.

Theoretical investigations and researches to develop conceptual framework and mathematical theory to model, analyse and control these structured, complex, large scale systems have been carried out since early sixties. Basically, the main idea behind this approach is the recognition that complex systems are structured in a hierarchical order. This approach in fact recommends that for a mathematical theory to claim to be dealing with large scale complex systems, the complexity of the real systems must be reflected in the structure of the model. Many research papers concerning the theory and methodology

of large scale systems were published throughout the sixties. In 1970, Mesarovic, Macho and Takahara presented one of the earliest formal quantitative treatments of hierarchical (multi-level) systems. Since then, a great deal of work has been done in this field and many papers which are based upon these theoretical works have been published and presented in many conferences from different fields. Many researchers have addressed themselves to various problems concerned with large scale systems. Excellent survey papers on the topic of hierarchical (multi-level) control and optimisation of large scale systems were given by Mahmoud (1977), Singh (1981,1982).

Roughly speaking, problems concerned with large scale systems may be divided into two broad areas: static problems and dynamic problems. However, this research is focussed only on hierarchical optimisation and control of large scale systems operating at steady state condition.

Because of economic reasons and intensive competition within the market sector, increasing pressure is stressed upon industry to improve efficiency and productivity of industrial plants. A method for the practical implementation of an on-line optimisation and control scheme for an industrial process considers the overall design as a two layer hierarchical system. This is a form of optimising control where the lower layer contains regulatory feedback controllers used to maintain the system at a steady state operating condition specified by controller set points. Optimisation is performed in the upper layer, where a steady state mathematical model is used to compute the optimal values of the controller set points to maximise the operational efficiency.

Recent techniques specifically consider a large scale industrial system as a collection of interlinked sub-systems. Modern computer technology which has resulted in low cost computer power may then be employed with the advantage in order to employ micro or mini-computers, implementated within a distributed hierarchical computer

structure, to coordinate, optimise and regulate individual decision task and sub-process. When designed carefully this control strategy has the advantages of reducing computer storage and computation time accomplished through parallel processing, increased system flexibility and reliability, and cheap hardware cost. Furthermore, it identifies and organises the information flow through the system. Thus provides effective use of feedbacks for control and decision making process which is attractive for on-line control purposes.

Three problems associated with on-line hierarchical optimisation and control of industrial processes are considered. Firstly, the behaviour of each process under control is not known and the mathematical model representing the real process is only an approximation. Secondly, the nature and magnitude of disturbances affecting the real process are stochastic. Lastly, timing problems arise because the real process, the sub-system decision units and the coordinator are processing at different speeds. One way to reduce the first two problems is to use feedback information from real process measurements. For timing problems, the local decision units at the infimal level should be synchronised with the coordinator and a simple synchronisation method - "semaphore" has been used.

Since late seventies, several hierarchical control schemes have been suggested for on-line control of steady state systems by Findeisen (1978,1979), Brdys (1978,1979), Roberts (1983), Shao (1983). These proposed schemes were investigated by off-line simulations of interconnected hierarchical systems. The objective of this research is to implement some of the proposed schemes through a distributed control system in order to investigate the closed-loop hierarchical control of a simulated interconnected industrial process. The on-line closed-loop hierarchical control schemes used in this research are the well-known Interaction Prediction Method and the Interaction Balance Method with local or global

feedback. Implementation problems that may arise for different closed-loop hierarchical control and optimisation of an interconnected system were examined. Study of on-line distributed hierarchical control with the local decision units operating asynchronisely had also been performed and implementated with the distributed hierarchical computer system.

The major contributions of knowledge of this thesis can be summarised as follows. This is the first time that hierarchical control algorithms (the Interaction Prediction Method and the Interaction Balance Method with local or global feeback) have actually been investigated in a real-time environment. With a suitable synchronisation scheme, parallel processing at the infimal level within the hierarchical structure has been achieved. This greatly reduces the computation time required by the local decision units which again reduces the overall computation time required.

Investigation of the problems relating to on-line hierarchical control and optimisation of interconnecting systems have been carried out. These problems are the transfer lags, measurement errors, subprocess and decision unit failures. An analogue computer was used to simulate these problems. The transfer lags were simulated by means of introducing a first order time constant in the controls and interaction variables. The measurement errors, subprocess and decision unit failures were simulated by varying the potentiometers within the analogue computer that correspond to the controls and the interaction variables. The stability, convergence and the system response of the decision problems when subjected to these problems were examined.

Coordinated by Interaction Prediction Method and the Interaction Balance Method with local feedback, simulation studies had been made concerning synchronisation of subsystem decision units and the effects of non-synchronism upon the stability and convergence of the

coordinator and local decision optimisation problems. Interaction Prediction Method with local feedback offers a stable and converging decision problems while the Interaction Balance Method with local feedback is very sensitive and becoming unstable when local decision units are operating asynchronously.

CHAPTER 2 HIERARCHICAL CONTROL AND OPTIMISATION OF LARGE SCALE SYSTEMS

2.1 Introduction

Hierarchical, multi-level system theory was introduced by Mesarovic and his associates (1970) with an objective of establishing a conceptual framework to mathematical theory in order to tackle complex multi-goal decision making systems. The basic idea behind this theory is the recognition that many real-life large scale complex systems are structured in a pyramid-like form. It is therefore natural to develop a mathematical model of the structural approach to control these complex systems.

Some key properties associated with hierarchical systems can be summarised as follows:

a) The decision making units are arranged in a pyramid-like structure;
b) Information flow between various levels of hierarchy are exchanged iteratively and (usually) vertically;
c) The objectives of local decision units may be in conflict which have to be resolved by the coordinator;
d) Time horizon increases as the level of hierarchy goes up;
e) Existence of a supremal unit.

Optimisation of large scale systems is motivated by the fact that there is a possibility of substantial economic savings which may happen in both the design and operational phases. Both static optimisation (finite dimensional) and dynamic optimisation (infinite dimensional) techniques are involved in the control of large scale systems.

An essential concept of the hierarchical control and optimisation approach for complex large scale systems control is the consideration of the overall control system as replaced by a set of smaller interdependent subsystems. Each subsystem has to serve a particular function, shares resources, and is governed by an interrelated objective and a set of constraints. Subsystems which are in the lowest (infimal) level of the hierarchy are often called infimal or local decision units while those in the higher (supremal) level are referred as coordinator or supremal units. The objectives of the subsystems in the infimal level are then solved independently under the intervention of a coordinator in the supremal level. The task of the coordinator is to account for the interconnections and constraints between the infimal units such that the overall system objective can be represented by the collection of individual subsystem objectives. Thus, the successful operation of hierarchical systems is best described by two processes: decomposition or infimal generation and coordination or overall objective synthesis.

Section 2.2 outlines the basic types of hierarchical structures. Sections 2.3 and 2.4 are concerned with alternative methods of coordination and optimisation respectively. Finally, a short summary is given in section 2.5.

2.2 Basic Types of Hierarchical Structures

Based on the hierarchical systems approach, an overall complex large scale problem may be decomposed into a collection of interconnected smaller subproblems arranged in a hierarchical structure. There are three basic types of hierarchical structure (Mesarovic 1970), namely,

1) Multi-strata hierarchical structure;
2) Multi-layer hierarchical structure;
3) Multi-echelon (Multi-level) hierarchical structure.

The classification of these hierarchical structures is based on the decomposition criterion chosen for the overall complex problem. It should be noted that many real-life complex systems may belong to more than one class of these hierarchical structures. Also, the operation of each of these types of hierarchy depends heavily on the information exchange mechanism between adjacent levels, the explicit specification of the subproblems and their objectives, and the proper manipulation of the subsystems activities. The hierarchical structures and their characteristics will be outlined in the following subsections.

2.2.1 Multi-strata Hierarchical Structure

The multi-strata type of hierarchical structure describes the system by a family of models each concerned with the behaviour of the system as viewed from a different level of abstraction. The levels of this type of structure are often called strata. For each level, there is a set of relevent features and variables, laws and principles in terms of which the behaviour of the system is described. It is necessary that the functioning on any level be as independent as possible of the functioning on other levels. The partitioning into strata has the objective of simplifying the overall complex problem by separating the problem into a number of smaller, better defined subproblems each of which is solved separately.

Let us illustrate the stratified description of a hierarchical structure by an automated production process with two strata shown in fig 2.1.

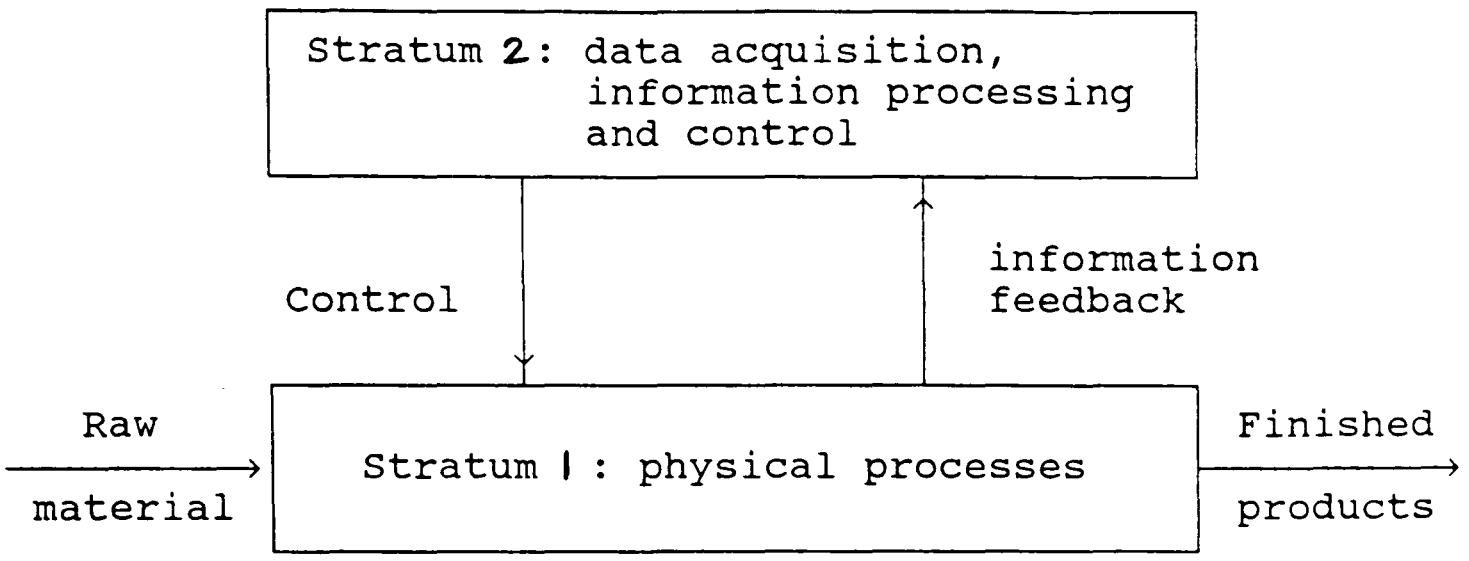


fig. 2.1 A two-strata diagram of an automated production process

These strata are used to deal with the same item, i.e. the finished product. On the first stratum, the product is viewed as a physical object to be changed in accordance with physical laws. On the second stratum which deals with data acquisition, information processing and control, the same item is viewed as a variable to be controlled and manipulated. A different model is used for each of these views of the system.

A Multi-strata hierarchical structure has the following characteristics,

a) The selection of strata depends upon the observer, his knowledge and interest in the operation of the system;
b) In general, contexts in which the operation of a system on different strata described are not mutually related; the principles used to describe the system on any stratum cannot generally be derived from the principles used on other strata;
c) There exists an interdependence between the functioning of a system on different strata;
d) Comprehension of a system increases by crossing the strata: the lower strata are assigned with more detailed and specialised descriptions while higher strata have a deeper understanding of its significance.

Decomposition on the basis of stratum is not based on the decomposition of a well-formulated mathematical programming problem. Therefore, there is no rigorous theory to justify its performance. This method partitions the control problem in such a way that it can be solved sequentially in strata with the result of one stratum serving as partial input to the other stratum below. Thus the decision making process possesses the characteristics of a staged process rather than those of a completely interacting one.

2.2.2 Multi-layer Hierarchical Structure

Multi-layer description of a hierarchical system is concerned with levels of decision complexity. The levels of this hierarchy are called layers. Essentially, the idea behind this hierarchical approach is that one defines a set of control problems whose solution is attempted in a sequentially manner. The overall control problem is substituted by a set of sequentially arranged simpler subproblems so that the solution of all the subproblems in the set implies the solution of the original problem. Within the hierarchy each layer operates at different time horizons, therefore, multi-layer description may be viewed as vertical decomposition of control structures. Multi-layer hierarchical system may further be divided into two different classes depending on the decomposition criteria, these are:

a) Functional multi-layer structure - decomposition according to control function;

b) Temporal multi-layer structure - decomposition according to time scale.

As an example of functional multi-layer hierarchies, we consider a functional four-layer control hierarchy under uncertainties. The four functional control layers are, namely, regulation, optimisation, adaptation and

self-organisation.

A schematic diagram of a functional four-layer control structure is shown in fig 2.2.

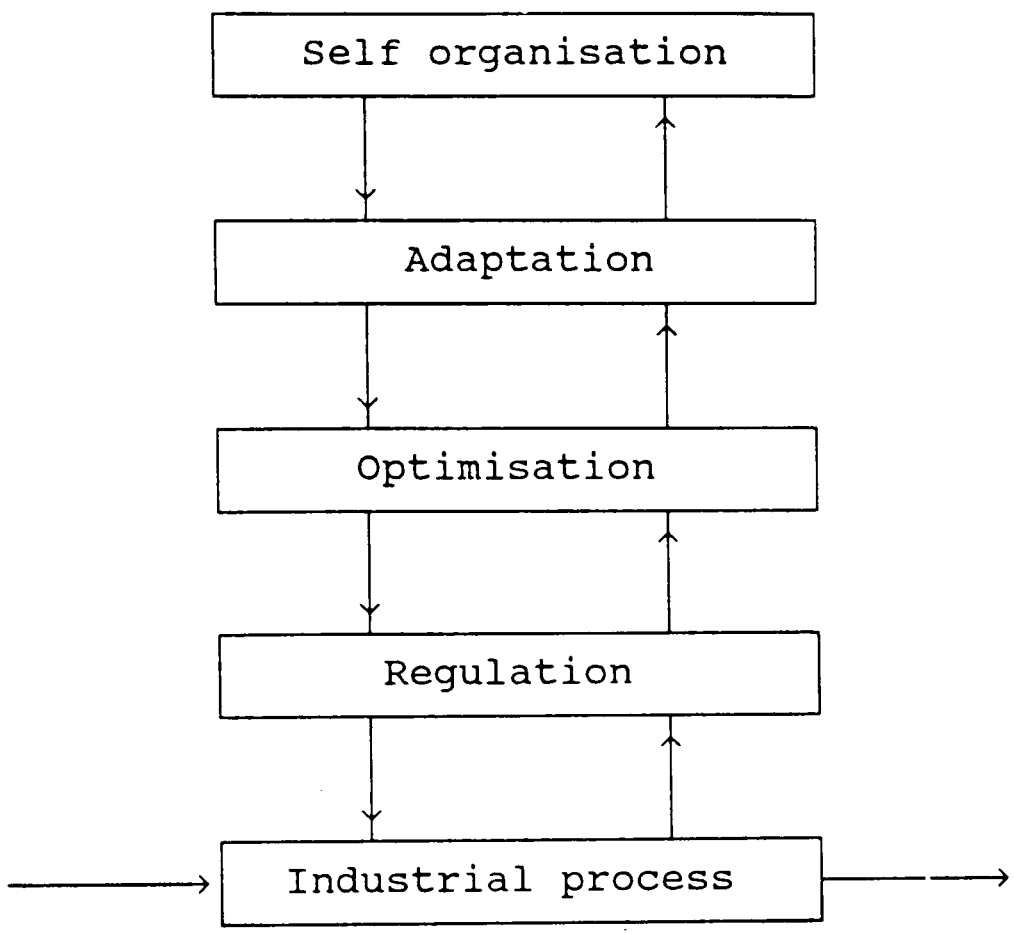


Fig.2.2 Schematic diagram of a functional four-layer control structure

Now let us look into the functional aspects of each layer within the hierarchy.

First layer: This is the regulation or direct control layer whose task is to maintain the process variables at prescribed set-point values under the influence of disturbances. This layer incorporates the functions of data acquisition, event monitoring and direct control.

Second layer: The optimisation or supervisory layer whose objective is to specify set-point values for the regulation layer. The set-point values are determined by optimising a mathematical model which approximates the real process under control.

Third layer: The adaptation or learning layer is concerned with specifying or updating the uncertain parameters of the mathematical model used by the supervisory layer in order to simplify the task of the second layer.

Fourth layer: The self-organising layer whose task is to select the structure, functions and strategies which are used on the lower layers so that an overall objective is being pursued as closely as possible. It can change the parameters of the models used by the lower layers if either the control action is unsatisfactory or the overall goal changes.

The characteristics of a functional multi-layer hierarchical structure are:

a) A natural hierarchy in which each layer has a priority of action over the layer below;
b) The layers of the hierarchy represent different kinds of control functions, therefore, require different kinds of information processing and computation algorithms;
c) The layers of the hierarchy can be designed to respond to disturbance inputs having different frequency characteristics.

Thus, functional multi-layer hierarchical approach provides a rational and systematic procedure for resolving complex large scale problems.

For temporal multi-layer control structure, the partition of the control problem into subproblems is based on different time scales relevant to the associated action functions. These time scales reflect the following factors:

(a) Minimum information acquisition time;
(b) Bandwidth properties or mean time between discrete changes in disturbance inputs;
(c) Minimum time horizon associated with the control action;
(d) Cost-benefit trade-off considerations.

Within a temporal multi-layer control structure, the i^{th} layer controller generates a control action, on an average, every T_i seconds, with $T_{i+1} > T_i$, i=1,2,.., based on

(a) Current input information;
(b) Targets and/or constraints provided by a $(i+1)^{th}$ layer controller;
(c) Feedback information provided by a $(k-1)^{th}$ controller.

The advantages of using temporal multi-layer decomposition approach for large scale system analysis are:

(a) Reducing the effects of uncertainty;
(b) Introduction of feedback of experience;
(c) Aggregating variables and simplifying models;
(d) Implementing systems integration through well-defined assignments of tasks and responsibilities.

An example of a temporal multi-layer structure considers a multi-time scale production planning and scheduling system whereby the planning may be on a yearly plan, monthly plan, daily schedule etc. Therefore each layer of the hierarchy generates the targets/constraints for the layer below and tries to maintain the prior plan or schedule using the feedback information.

2.2.3 Multi-echelon (Multi-level) Hierarchical Structure

This is the most general type of hierarchy which uses a structural or organisational approach to decouple a

system. The basis of this hierarchical decomposition considers the partitioning of a system into separate subsystems each with its own and perhaps conflicting goals and interaction among the subsystems. The subsystems are positioned on different levels within the hierarchy such that each one can coordinate lower level units and be coordinated by a higher level one. Each subsystem pursues its own assigned goal independently. Since the subsystems are coupled and interacting, a higher level unit (coordinator) is used to coordinate the subsystems in the lower level to account for the disturbances introduced by the interacting subsystems. The levels of this hierarchical structure are called echelons.

Basically, there are three categories concerned with decision making systems with respect to hierarchical arrangements of decision units. They are single-level single-goal systems, single-level multi-goal systems and multi-level multi-goal systems which are shown in fig.2.3.

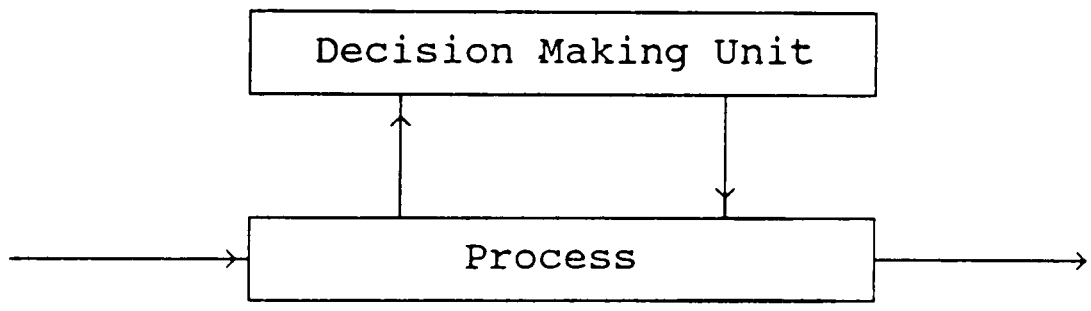


Fig.2.3a Single-level single-goal system

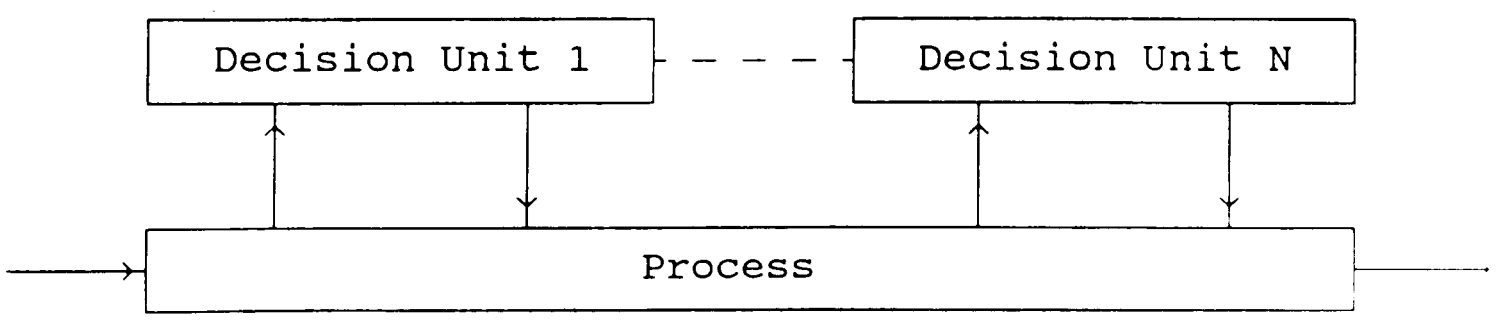


Fig.2.3b Single-level multi-goal system

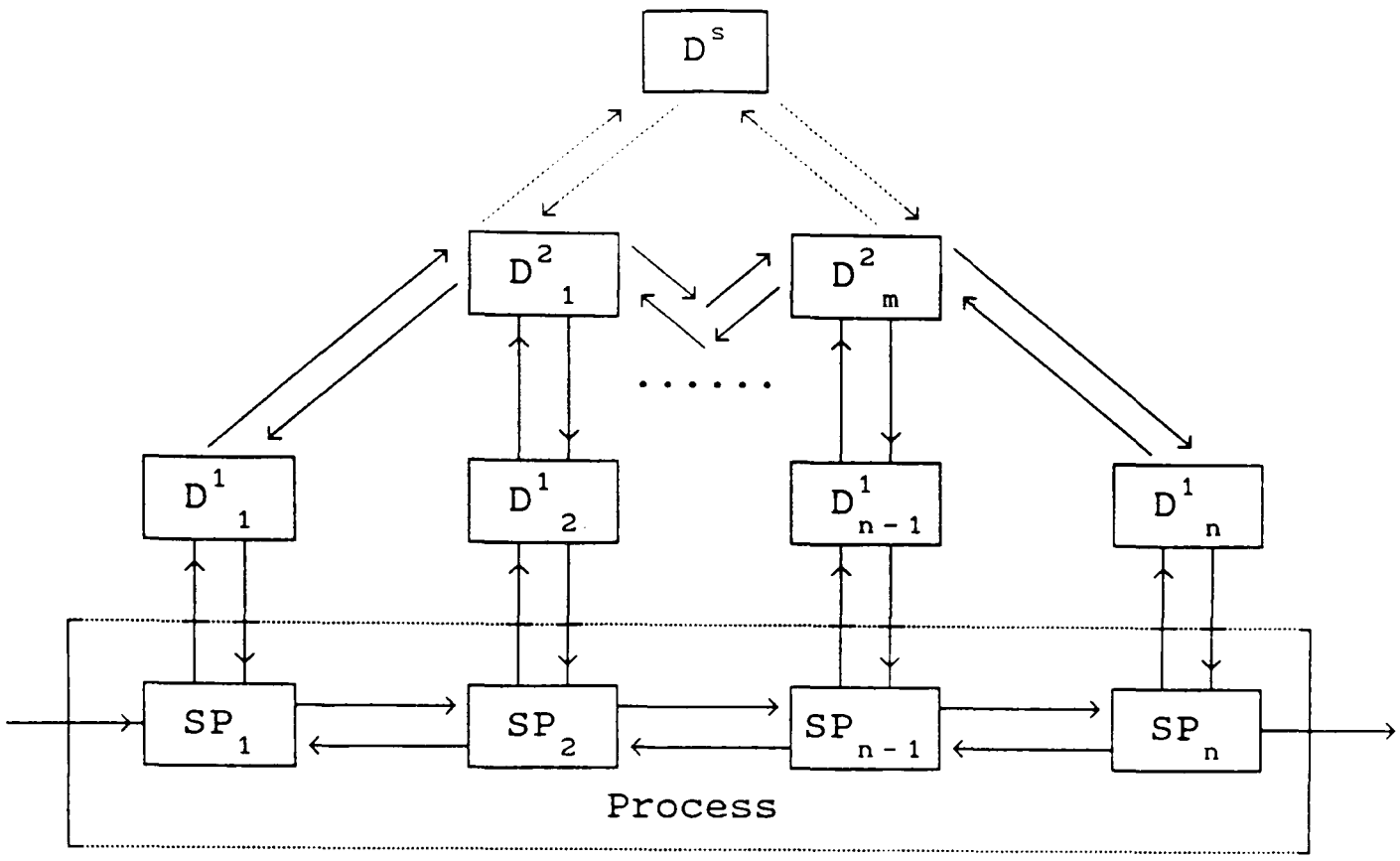


Fig.2.3c Multi-level multi-goal system

For single-level single-goal system, a goal is defined for the overall system, and all decision variables are selected in order to satisfy this goal. This centralised decision making system has an advantage of conceptual simplicity but technically it is very difficult to be implemented for large scale complex problem.

Single-level multi-goal system consists of a family of decision units, each with its own goals. The goals are not necessarily conflicting. This is a decentralised decision making system with the advantage of problem sharing.

Multi-level multi-goal system is the general representation of large scale system where the decision units are arranged in a pyramid-form structure. Each level consists of a number of decision units, each with its own goal. Some of these decision units are coordinated by another decision unit in the level above. There exists a supremal unit at the top level which characterised this decision making system.

The multi-level approach for the decomposition of large scale systems has the following advantages:

(a) Reduction in development costs;
(b) Reduction in total computation effort;
(c) Reduction in data/information exchange;
(d) Increase system reliability.

2.3 Coordination Methods

In order to solve complex large scale systems, it is necessary to decompose the system and/or its control into a set of smaller subsystems, and in addition to provide a mechanism for coordination of the subsystems so as to achieve the overall system objective. The transformation of a given integrated system into a hierarchical one can be achieved by different ways as outlined in the preceding sections. However, the problem of coordinating these subsystems is not always straight forward and places a practical limitation on the decomposition. This is especially true in an on-line system because the response constraint in real time may hinder many coordination algorithms which are theoretically possible (e.g. unfeasible methods). Therefore it is important to determine techniques for coordination in each type of multi-level structures and to consider the methods by which each structure handles physical disturbances in the system together with interaction disturbances introduced by the decomposed nature of the system.

Coordination between the supremal and the infimal levels within the hierarchy can be achieved by transmitting coordination signals from the supremal coordinator to each infimal unit and, in turn, receives performance information from the infimal units. Most of coordination schemes are essentially a combination of two distinct approaches:

Interaction Balance Method (IBM) and Interaction Prediction Method (IPM). These coordination method are similar in the sense that they initiate local decision optimisation computation in the infimal level by specifying intervention parameters from the supremal level. The method by which the supremal level calculates the coordination parameters and intervenes the infimal level defines the coordination strategy.

In the following subsections, these coordination methods are briefly described. A detailed treatment on on-line coordination methods suitable for closed-loop control and optimisation of interconnected system can be found in chapter 5.

2.3.1. Interaction Prediction Method (IPM)

This coordination method is also known as model coordination, primal coordination, method of projection and parametric decomposition.

The main feature of this method is that the supremal unit prescribes the interaction variables of the subsystems. At each iteration, the coordinator specifies the interaction variables, and the infimal units proceed to solve their model based local decision problems on the assumption that the interaction variables are as exactly as predicted by the supremal unit. Based on the measurement taken from the real process (real interaction variables) and the solution information from the local decision units, the coordinator employs an iterative procedure which adjusts the specification of the interaction variables until the global optimum is obtained, i.e. the real interaction variables are identical to the predicted ones.

Interaction Prediction method is inherently suitable for on-line application since all the intermediate results of the iterative optimisation can be applied directly to

the real process because all interconnection contraints are always satisfied. Hence, this method is classified as feasible coordination strategy.

One of the disadvantages of this coordination method is the overdetermined subsystem optimisation problems at the infimal level. This overdetermined problem can be eased by either reducing the effective number of interconnections between subsystems or relaxing the interconnection constraints on the expense of losing the feasibility property which this coordination method posesses. Another source of difficulty is that, in general, the gradient of the objective function is not easy to compute and may not even exist. Therefore, this is not really possible to solve the coordinator problem effectively without gradient information. Furthermore, it is important to ensure that the coordination variables are such that the local decision solutions lie in the feasible region within the constraint boundaries.

2.3.2 Interaction Balance Method (IBM)

Alternative names of this coordination method are goal coordination, price coordination and dual coordination.

This coordination method considers the interconnections between subsystems as additional constraints on the local optimisation problem. Suitable modification of the infimal objective functions is made to take account of the interactions between subsystems. A decomposable Lagrangian may be formed to provide the modified objective function for each subsystem optimisation problem. The task of the coordinator is to select interaction inputs to the infimal units such that all interconnection constraints are satisfied (i.e. in balance). This means that the final optimum solution of the overall system is obtained.

Since the interconnection constraints are not satisfied during the early stages of the iterative optimisation procedure and only the final 'balanced' solution can be applied to the real process, interaction balance method is termed as non-feasible coordination strategy.

Analytically, coordination by Interaction Balance Method is more appealing than the Interaction Prediction Method as the formulation of the former method is based on the well-known Lagrangian theory. Only mild assumptions of continuity and convexity are needed to ensure existence of solution of the optimisation problem. If the solution of the Lagrangian function is unique, the gradient of the dual function exists. Then reasonably fast gradient-type optimisation algorithm can be implemented to solve the dual (coordinator) problem. However, if the problem is not convex, duality gap problem may occur. Then the solution obtained by the dual formulation may or may not be the correct optimum.

2.4 Optimisation Methods

Having decomposed the overall control problem into subproblems, optimisation is performed in both the infimal and supremal levels within the hierarchy. The local decision units compute the optimal controller set-point values for the subprocesses under control and the coordinator determines the intervention parameters to account for the constraints and interaction between subsystems such that the overall performance requirements are achieved.

A number of numerical methods based on mathematical programming are available for solving these optimisation problems. The choice of the method depends upon its

numerical properties and the nature of the problem under investigation. The optimisation problem may be linear or non-linear, unconstrained or constrained, mono-variable or multi-variable. The numerical properties of an optimisation algorithm are the existence of the numerical solution approach, the convergence of the algorithm and computation time required. For steady-state optimisation problems, direct and gradient methods are commonly used for unconstrained problems. The simplex method is ideal for linear constrained problems. Primal and dual methods incorporating augmented Lagrangians may be employed for non-linear constrained optimisation problems.

Details of these optimisation algorithms are well documented in the mathematical programming literature. Each has its advantages and disadvantages. Enough experience with these numerical algorithms enables proper selection of algorithm or combination of algorithms to suit the nature of a particular problem.

2.5 Summary

This chapter introduces the basic concept of hierarchical control and optimisation of large scale systems using the decomposition and coordination approach. Decomposition of complex system into one or a mixture of the three basic types of hierarchical structure, namely, the muti-strata, multi-layer, and multi-level, depends on the nature of the system under study. The characteristics of the three basic hierarchical structures have been briefly described. Then, methods of coordinating the decomposed system using the Interaction Prediction and the Interaction Balance principles are briefly outlined. Finally, a brief description of optimisation methods that may be employed for system optimisation is included.

CHAPTER 3 DISTRIBUTED HIERARCHICAL COMPUTER SYSTEM (DHCS)

3.1 Introduction

Within the Computer Control Laboratory of City University there is a distributed two-level hierarchical computer system. The infimal level of the hierarchy contains four I-MIC micro-computers and a LSI-11/02 mini-computer, which are used as local decision units for direct control of pilot plants and a small analogue computer system operating at steady state conditions. A DEC LSI-11/23 mini-computer with a time shared operating system is employed at the supremal level which acts as a host machine to coordinate and supervise computers (local decision units) at the lower level. There is also a BBC micro-computer system with a visual display, a disk drive and a printer.

At present the I-MICs control a pilot scale freon vaporiser, a mixing process and simulated interconnected dynamic processes on the analogue computer. The LSI-11/02 controls a pilot-scale eight zone electrically heated travelling load furnace. A schematic diagram of the distributed hierarchical computer system is shown in fig.3.1

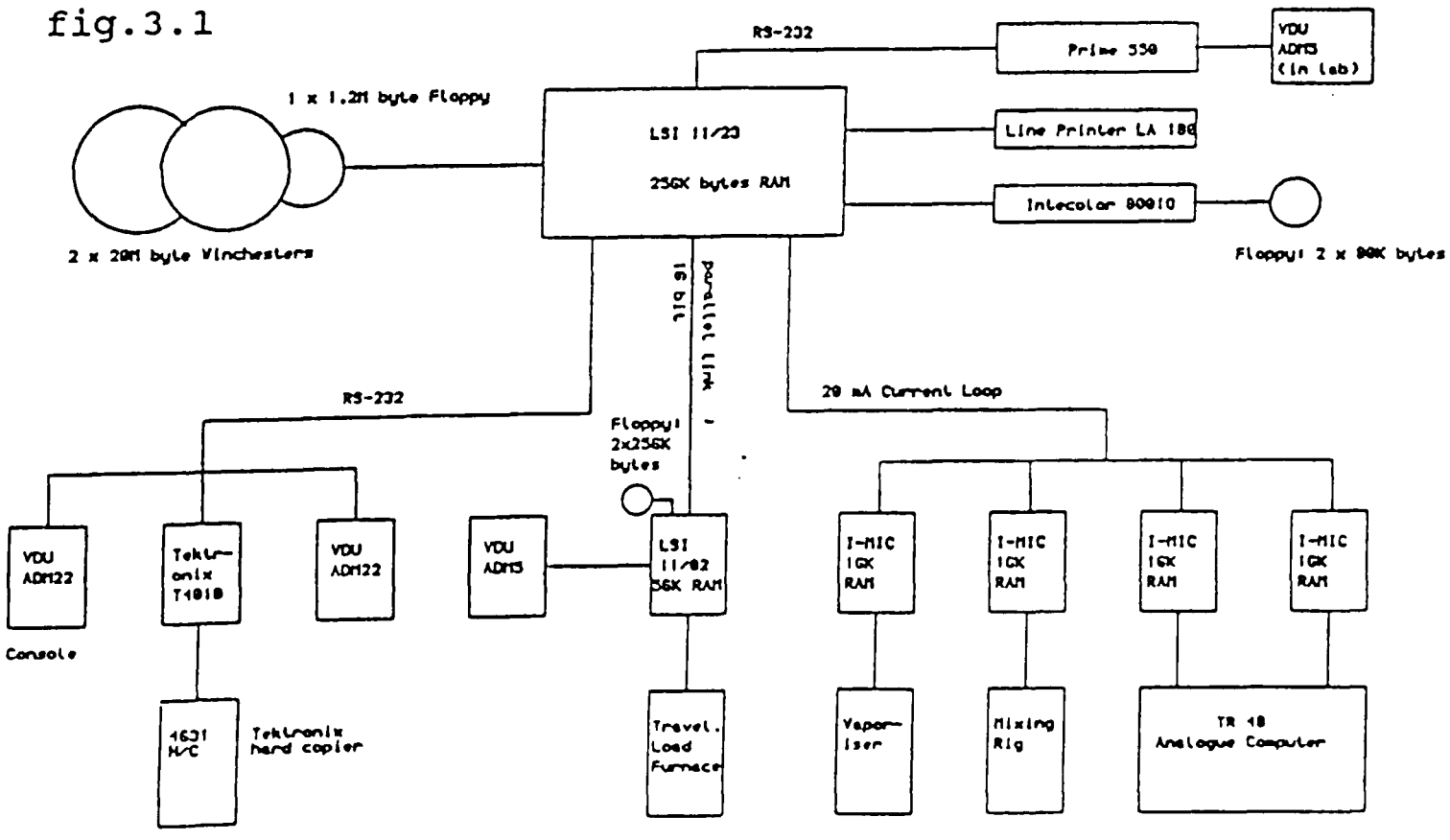


Fig.3.1 Schematic diagram of the distributed hierarchical computer system

In order to investigate the decomposition-coordination methods used for on-line optimisation and control of interconnected system a two-level hierarchical structure was established. At the infimal level of the hierarchy two I-MIC micro-computers are used as local decision units whose task is to determine the optimal controller set point values from a given set of coordinator intervention variables. The LSI-11/23 mini-computer which acts as the coordinator supervises the I-MICs in order to achieve the overall optimal solution. The coordinator is also used to synchronise the local decision units before optimal controls are applied to the simulated interconnected dynamic real process.

A EAL Pace TR48 analogue computer is used to simulate an interconnected dynamic real process with first order transfer lags introduced in the controls and interconnected inputs. The existence of disturbances and model-reality differences cause the real process under control to deviate from its desired optimal steady state condition. Corrective action can be made using feedback based on measurements by which the current state of the real process is inferred relative to an assesssment of the desired state. Feedback information from the analogue computer is implemented globally or locally within the hierarchy. The two level hierarchical structure is shown in fig.3.2.

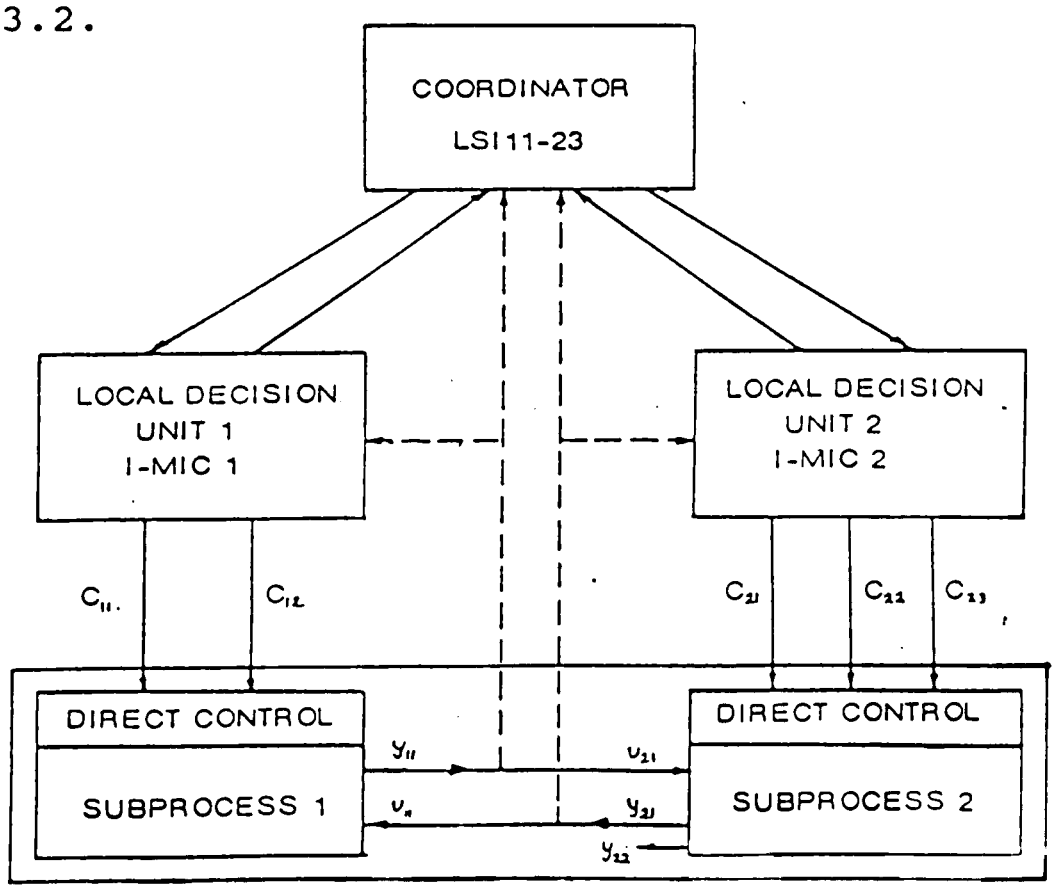


Fig.3.2 A two level hierarchical structure

A brief description on the computers used as coordinator, the local decision units and the simulated real process, i.e. the LSI-11/23, the I-MIC and the TR48, can be found in the following sections.

3.2 Coordinator (LSI-11/23)

The LSI-11/23 has a full complement of 256K bytes RAM and runs the TSX-PLUS time shared operating system. It is a general purpose time shared operating system which can provide computing facilities, both off-line and real-time, for up to twenty users working concurrently. The TSX-PLUS operating system is based on the DEC RT-11 single job monitor with some extended features such as a transparent line-printer spooling system, a shared file record locking facility, an inter-job message communication facility, a program performance monitor system, command files with parameters and a logon and usage accounting system. Such features greatly enhance the system performance, particularly in real-time applications. Having logged on, each user is allocated with 20K bytes of memory which can be extended up to 56K bytes. This is more than the previous single job operating system RT-11 can provide.

Programs and data are stored on two 20M bytes Winchester disks and a 1.2M bytes floppy disk drive. The peripherals which are available to the LSI-11/23 include one ADM5 and two ADM22 visual display units, a Tektronix (T4010) storage tube graphics terminal, a hard copy unit and an Intecolor (8001G) colour graphics terminal. The latter is used to display the current measurements from any process on a mimic diagram.

Although several high-level languages (BASIC, FORTRAN, PASCAL, COBOL-PLUS, APL etc.) are available for programming on the LSI-11/23, FORTRAN is used since, at present, this is the only one which can provide the real-time support required (timer functions, interrupt handling etc.).

3.3 Local Decision Unit (I-MIC)

An I-MIC is a micro-processor based industrial controller by KRATOS Instem Limited which may be programmed either in interpretive CONTROL BASIC or in machine code via the I-MIC monitor. An Intel 8085 micro-processor is the heart of this industrial controller. The controller uses a 10 inch rack-mountable bin where the processor module (F030), memory module (F043), removable power supply module, industrial interface (data highway) and input/ouput modules are resident.

The F030 processor module has an on-board memory capacity of 2K bytes of static CMOS RAM and 8K bytes of EPROM. The first 4K bytes of EPROM contain the CONTROL BASIC interpreter, the monitor occupies the next 2K bytes with the last 2K bytes unused. Two serial data communication links are available on this board. An Intel 8251 USART (Universal Asynchronous Receiver/transmitter) driven serial output port, interfaced with a RS232 serial link cable, is used to connect a teletype. The other serial communication link is software driven which links the I-MIC and the LSI-11/23 via a 20mA current loop cable.

The objective of the F043 memory module is to extend the memory capacity of both the RAM and EPROM area by 16K bytes. This module also has on-board programming facilities using different accessory modules. Both the F030 and F043 are DEC quad-height modules.

The I-MIC can accommodate up to 14 industrial input/output modules of different functions. These are all "memory mapped" modules which communicate with the CPU through the industrial interface. The following modules are used for transmitting and receiving data from the simulated real process:

G230 : This is a 2 channel DAC interface module which converts the digital output from the I-MIC to analogue signal usable by the analogue computer.

G226/G266 : This is a 12 bit ADC/16 channel multiplexer interface module which is used to convert the output from the analogue computer to digital form ready to be used by the I-MIC.

3.4 Simulated Interconnected Process (TR48)

The computer used to simulate the interconnected industrial process is a EAL Pace TR48 general purpose analogue computer which is composed of solid state computing components. It is of modular design with eight different computing components : operational amplifiers, dual integrators, quarter-square and bi-polar multipliers, x^2 diode function generators, log x and 1/2 log x diode function generators, sine-cosine diode function generator, variable diode function generators and signal comparators.

The front of the analogue computer contains three panels, a removable pre-patch panel, a monitoring and control panel and a panel containing attentuators and function switches. Each component module is pre-wired to accept a combination of computing components so that the computer configuration can be altered very easily. Located to the left of the pre-patch panel is the monitoring and control panel. A digital voltmeter, a multi-range voltmeter and a push button signal selector provide the necessary monitoring facilities. The control section contains a power switch, a computer mode of operation selector and a pre-patch panel engaging and disengaging switch. The attentuators and the function switches panel is situated to the right of the pre-patch panel. There are fifty coefficient potentiometers and five function switches mounted on this panel. All the potentiometers and function switches are terminated on the pre-patch panel.

There are also three trunk patch areas on the pre-patch panel, each of which is terminated at a connector on the rear of the TR48. These connectors can be used to link

external equipment such as an oscilloscope and x-y plotter.

3.5 Summary

The distributed hierarchical computer system used to investigate on-line closed-loop hierarchical control and optimisation of an interconnected process has been outlined. A brief description of various components of the computer system, namely, the LSI-11/23 mini-computer, the I-MIC industrial micro-controller and the TR48 analogue computer that formed the hierarchical two level structure are included in this chapter.

CHAPTER 4 FORMULATION OF CONTROL PROBLEM

4.1 Control Problem Formulation

This chapter is concerned with the mathematical formulation of the steady state control problem. Particular attention is paid to the formulation of subsystems and system equations, the kinds of constraints imposed on the system, and the formulation of the performance index of the system.

Using the hierarchical control approach, the overall control system is decomposed into an assemblage of interconnected subsystems. Each subsystem has its own goal and optimisation task. The i^{th} subsystem is shown in fig.4.1,

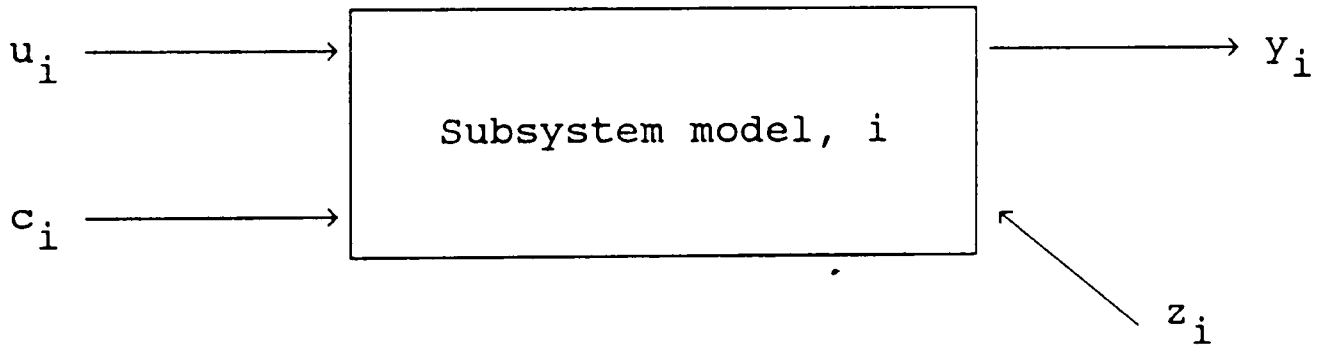


Fig.4.1 The i^{th} subsystem model

where the vectors c_i, u_i, y_i and z_i are the i^{th} subsystem control interaction input, interaction output and external disturbances with appropriate dimension respectively. $c_i \in \mathcal{C}_i$, $u_i \in \mathcal{U}_i$ and $y_i \in \mathcal{Y}_i$, where $\mathcal{C}_i$, $\mathcal{U}_i$ and $\mathcal{Y}_i$ are finite dimensional spaces. It is assumed that external disturbances affecting the system are constant over the considered time interval and therefore can be omitted from the system equations. We assume that each subsystem, including its control system, is described by,

$$y_i = F_{*i}(c_i, u_i) \qquad i \in \overline{1,N} \tag{4.1}$$

The subscript "*" denotes all mapping related to the real subsystems (not models). The i^{th} subsystem input-ouput mapping is defined as,

$$F_{*i} : \mathcal{C}_i \times \mathcal{U}_i \longrightarrow \mathcal{Y}_i \qquad i \in \overline{1,N} \tag{4.2}$$

The interconnection between subsystems are defined by the linear coupling equations,

$$u_i = H_i\, y = \sum_{j=1}^{N} H_{ij}\, y_j\ , \qquad i \in \overline{1,N} \tag{4.3}$$

We denote $c \triangleq (c_1,\ldots,c_N) \in \mathcal{C}_1 \times\ldots\times \mathcal{C}_N \triangleq \mathcal{C}$

$u \triangleq (u_1,\ldots,u_N) \in \mathcal{U}_1 \times\ldots\times \mathcal{U}_N \triangleq \mathcal{U}$

$y \triangleq (y_1,\ldots,y_N) \in \mathcal{Y}_1 \times\ldots\times \mathcal{Y}_N \triangleq \mathcal{Y}$

where H_i, H_{ij} are the interconnection matrices composed of zeros and ones.

Then the subsystem equations and system structure equations can be written jointly as,

$$\left.\begin{array}{l} y = F_*(c,u), \qquad u = Hy\ ; \\ \\ F_* : \mathcal{C} \times \mathcal{U} \longrightarrow \mathcal{Y}, \\ \\ F_*(c,u) \triangleq \begin{bmatrix} F_{*1}(c_1,\ u_N) \\ \vdots \\ F_{*N}(c_1,\ u_N) \end{bmatrix}\ ;\quad H \triangleq \begin{bmatrix} H_1 \\ \vdots \\ H_N \end{bmatrix} \end{array}\right\} \tag{4.4}$$

It is assumed that for each overall control vector $c \in \mathcal{C}$ applied to the system, there exists a unique overall output vector y. Thus the system as a whole may be described by the mapping,

$$K_* : \mathcal{C} \longrightarrow \mathcal{Y} \tag{4.5}$$

$$\text{i.e.} \qquad y = K_*(c) = [K_{*_1}(c), \ldots, K_{*_N}(c)] \tag{4.6}$$

Since the (real) system relationships are not known exactly, mathematical models are used to approximate the real subsystems under consideration. Similar to the real system relationships as shown in equations (4.1), (4.2), (4.5) and (4.6) the following model equations can be written as,

$$y_i = F_i(c_i, u_i), \qquad i \in \overline{1,N} \tag{4.7}$$

$$F_i : \mathcal{C}_i \times \mathcal{U}_i \longrightarrow \mathcal{Y}_i \tag{4.8}$$

where F_i is the i^{th} subsystem model input-output mapping,

$$y = F(c,u), \qquad i \in \overline{1,N} \tag{4.9}$$

$$F: \mathcal{C} \times \mathcal{U} \longrightarrow \mathcal{Y} \tag{4.10}$$

where F is the system model input-output mapping,

$$y = K(c) \tag{4.11}$$

$$K : \mathcal{C} \longrightarrow \mathcal{Y} \tag{4.12}$$

where K is the model output mapping corresponding to equation (4.5) of the real system mapping.

Having formulated the subsystem and system equations, we look into the kinds of constraints imposed on the subsystems. It is assumed that the local constraints are given explicitly as,

$$(c_i,u_i) \in CU_i = \{(c_i, u_i) \in \mathcal{C}_i \times \mathcal{U}_i : G_{ip}(c_i;u_i) \leq 0,\ p \in P_i\},$$

$$i \in \overline{1,N} \tag{4.13}$$

where G_{ip} is the local constraint mapping defined by

$$G_{ip} : \mathcal{C}_i \times \mathcal{U}_i \longrightarrow R^{m_i} \tag{4.14}$$

and P_i denotes a set of integer indices.

If output variables are also involved in the local constraint set, equations (4.13) and (4.14) will be in the form,

$$(c_i, u_i, y_i) \in CUY_i = \{(c_i, u_i, y_i) \in \mathcal{C}_i \times \mathcal{U}_i \times \mathcal{Y}_i : G_{ip}^{y}(c_i, u_i, y_i) \leq 0, p \in P_i\},$$

$$i \in \overline{1,N} \tag{4.15}$$

$$\text{and} \quad G_{ip}^{y} : \mathcal{C}_i \times \mathcal{U}_i \times \mathcal{Y}_i \longrightarrow R^{m_i} \tag{4.16}$$

where the mappings G_{ip} and G_{ip}^{y} are assumed to be known exactly. Consequently, the local constraints can be written joinly in the form

$$(c,u) \in CU = \{(c,u) \in \mathcal{C} \times \mathcal{U} : G(c,u) \leq 0\}, \tag{4.17}$$

$$(c,u,y) \in CUY = \{(c,u,y) \in \mathcal{C} \times \mathcal{U} \times \mathcal{Y} : G^{y}(c,u,y) \leq 0\}. \tag{4.18}$$

as before.

Finally, we assume that each subsystem is associated with a scalar value of performance index expressed explicitly in (c_i, u_i). The subsystem performance indices have the form,

$$Q_i : \mathcal{C}_i \times \mathcal{U}_i \longrightarrow R^1, \qquad i \in \overline{1,N} \tag{4.19}$$

If output is involved, the performance indices become

$$Q_i : \mathcal{C}_i \times \mathcal{U}_i \times \mathcal{Y}_i \longrightarrow R^1, \qquad i \in \overline{1,N} \tag{4.20}$$

Accordingly, we can assume the overall performance index to have the form

$$Q = \Psi\ (Q_1, \ldots, Q_N) \tag{4.21}$$

where $\Psi : R^N \longrightarrow R^1$ is some strictly order perserving mapping. e.g., Ψ is of additive form, i.e.

$$Q = \sum_{i=1}^{N} \{\ Q_i\ \} \tag{4.21a}$$

Finally, we can incorporate the global control problem into the overall system performance index. The task of the global control problem is to optimise the overall system performance index subjected to the constraints, i.e.,

$$\begin{aligned} &\min_c \{ \sum_{i=1}^{N} Q_i\ (c_i, u_i) \} \\ &\text{s.t.} \quad y_i = F_i\ (c_i, u_i)\ , \qquad i \in \overline{1,N} \\ &\qquad G_{ip}\ (c_i, u_i) \le 0\ , \qquad p \in P_i \\ &\qquad u_i = \sum_{j=1}^{N} H_{ij}\ y_j \end{aligned} \tag{4.22}$$

If output is involved, the task of the overall control problem becomes

$$\begin{aligned} &\min_c \{ \sum_{i=1}^{N} Q_i\ (c_i, u_i, y_i) \} \\ &\text{s.t.} \quad y_i = F_i\ (c_i, u_i, y_i)\ , \qquad i \in \overline{1,N} \\ &\qquad G_{ip}^{y}\ (c_i, u_i) \le 0\ , \qquad p \in P_i \\ &\qquad u_i = \sum_{j=1}^{N} H_{ij}\ y_j \end{aligned} \tag{4.23}$$

4.2 Mathematical Model of the Hierarchical Control Structure

A steady state system model consisting of two interconnected subprocesses, simulated by the analogue computer, is used to investigate different coordination methods for closed-loop hierarchical control. The computer system structure used for simulation is shown in fig.3.2.

First order time constants are introduced to the interaction inputs and the controls. The mathematical model of the subprocess output equations is:

$$\begin{bmatrix} y_{11} \\ y_{21} \\ y_{22} \end{bmatrix} = \begin{bmatrix} f_{11}\,(\underline{c}_1,\ \underline{u}_1) \\ f_{21}\,(\underline{c}_2,\ \underline{u}_2) \\ f_{22}\,(\underline{c}_2,\ \underline{u}_2) \end{bmatrix} = \begin{bmatrix} c_{11} - c_{12} + 2u_{11} \\ c_{21} - c_{22} + u_{21} \\ 2c_{22} - c_{23} - u_{21} \end{bmatrix}$$

The performance indices are:

$$Q_1\ (\underline{c}_1, \underline{u}_1, \underline{Y}_1) = (y_{11}-1)^2 + c_{11}^2 + c_{12}^2$$

$$Q_2\ (\underline{c}_2, \underline{u}_2, \underline{Y}_2) = 2(y_{21}-2)^2 + (y_{22}-3)^2 + c_{21}^2 + c_{22}^2 + c_{23}^2$$

The reality output equations are:

$$\begin{bmatrix} y_{*11} \\ y_{*21} \\ y_{*22} \end{bmatrix} = \begin{bmatrix} f_{*11}(\underline{c}_1, \underline{u}_1) \\ f_{*21}(\underline{c}_2, \underline{u}_2) \\ f_{*22}(\underline{c}_2, \underline{u}_2) \end{bmatrix} = \begin{bmatrix} 1.4c_{11} - 0.6c_{12} + 1.8u_{*11} \\ 1.3c_{21} - 1.1c_{22} + 1.1u_{*21} \\ 2.3c_{22} - 0.7c_{23} - 1.1u_{*21} \end{bmatrix}$$

The system constraints are:

$$\{g_1(\underline{c}_1, \underline{u}_1, \underline{y}_1) \in R^4 : |c_{11}| \leq 1,\ |c_{12}| \leq 1,\ y_{11} \geq 0,\ (0.8 - c_{12} - 0.6u_{11}) \geq 0\}$$

$$\{g_2(\underline{c}_2, \underline{u}_2, \underline{y}_2) \in R^5 : |c_{21}| \leq 1,\ |c_{22}| \leq 1,\ |c_{23}| \leq 1,\ y_{21} \geq 0,\ y_{22} \geq 0\}$$

The coupling equation is :

$$\begin{bmatrix} u_{11} \\ u_{21} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_{11} \\ y_{21} \\ y_{22} \end{bmatrix}$$

The mathematical model of the interconnected subprocesses is taken from Roberts P.D. (1982).

4.3 Summary

Mathematical formulation of large scale control problem using the hierarchical system approach was presented in this chapter. The original control problem, with a pre-defined overall performance index and system constraints, was decomposed into a collection of smaller scale subproblems. Then, the model of each subproblem was formulated with the task to optimise its performance index subjected to the constraints imposed. Once all the subproblems were formulated, the global control problem can then be formed by assuming that the overall performance index was the sum of performance indices of the subproblems. Finally, the steady state system model consisting of two interconnected subprocesses, simulated by an analogue computer, was included.

CHAPTER 5 ON-LINE COORDINATION METHODS

5.1 Introduction

Different coordination methods have been suggested and developed in the last two decades for steady state optimisation and control of large scale systems (Mesarovic, 1970; Findeisen, 1978). Basically, there are two principal coordination methods, namely, the Interaction Prediction Method (IPM) and Interaction Balance Method (IBM). These coordination methods are similar in the sense that they start the local decision optimisation problem in the lower level by specifying intervention parameters from the supremal level. The method by which the supremal level calculates the coordination parameters and intervenes the infimal level defines the coordination strategy. A variety of feedback schemes have been suggested for on-line optimising control using these coordination methods. Feedback informations from the simulated real process used by the coordinator (global feedback) or local decision units (local feedback) have been implemented using the distributed hierarchical computer system to study the on-line coordination method - the interaction balance coordination and interaction prediction coordination strategy.

Section 5.2 describes the closed-loop coordination methods. Examination of these methods for on-line closed-loop control of simulated interconnected industrial processes has been performed and difficulties encountered during the investigation will be discussed in the subsequent sections.

5.2 Closed-loop Coordination Methods

Using open-loop implementation of coordination methods relies on accurate mathematical models which truly

represent the steady state behaviour of the industrial subprocesses under control. In practice, model-reality differences often occur. Feedback measurements from the real process may be used in an attempt to improve performance of the system and account for the model-reality difference. There are two principal methods in which feedback information from the real process may be employed. Global feedback is the scheme where the process measurements are sent to the coordinator; and the process measurements used in local decision units are known as local feedback scheme. The following subsections describe the implementation of the global feedback and local feedback schemes using the interaction prediction method and interaction balance method. Details concerning with the existence, solvability, convergency of the coordination methods can be found in Findeisen (1980).

5.2.1 Interaction Prediction Method with Global Feedback (IPMGF)

Feedback information is used by the coordinator in this coordination strategy which has been investigated by Findeisen (1974) and Wozniak (1976). The local decision problems are computed on the basis of the subsystem models. Hence, local decision problems remain the same as the open-loop. The local optimisation problem (LOP) is as follows:

LOP —

For a given coordination variable $v \in \mathcal{V}$,
find the control

$$\hat{c}_i(v) = \underset{C_i(v)}{\arg\min}\, Q_i(., H_i v) \qquad (5.1)$$

where $C_i(v) \triangleq \{c_i \in \mathcal{C}_i : (c_i, H_i v) \in CU_i \wedge v_i = F_i(c_i, H_i v)\}$

Since the coordination variable specifies the

subsystem outputs and interactions, i.e $y_i = v_i$ and $u_i = H_i v$ are strict equalities in the local optimisation problem (LOP), therefore the existance of LOP solution depends on the assumption of non-empty local feasible set V,

$$v \in V_0 \triangleq \{v \in \mathcal{Y} : \forall_i \in \overline{1,N} \quad C_i(v) \neq 0\} \tag{5.2}$$

Except in special cases, the set V_0 is very difficult, if not impossible, to determine.

Once the LOP is solved, the optimal controls are then applied to the real process and the measurements are transmitted to the coordinator. The coordinator solves the on-line problem based on the information from the local decision units and the measurements from the real process. The task of the coordinator (COP) is to minimise the real performance index Q within the feasible set V_0,

COP

Find a coordination variable

$$\hat{v} = \arg \min_{V_* \cap V_0} Q\,(\hat{c}(.),\ HK_*(\hat{c}(.))) \tag{5.3}$$

where $V_* \cap V_0 \triangleq \{v \in \mathcal{Y} : K_*(\hat{c}(v)) \in Y \wedge \forall_i \in \overline{1,N}\ (\hat{c}_i(v), H_i K_*(\hat{c}(v))) \in CU_i\} \cap V_0$,

$\hat{c}(v) \triangleq (\hat{c}_1(v), \ldots, \hat{c}_N(v)), \quad i \in \overline{1,N}$

$K_*(c(v))$ represents the real system outputs.

5.2.2 Interaction Prediction Method with Local Feedback (IPMLF)

This coordination method was first studied by Tatjewski and Cygler (1980) where feedback information from

the real process is used by the local decision units. Completely decentralised information structure can be achieved since each local decision unit solves its decision problem based on information feedback from its own subprocess only. The basic idea of this coordination strategy is to use a 'shift vector s' which forces the model interaction output vector to be equal to the real interaction output vector. The iterative mechanism of this coordination method operates as follows: the coordinator specifies the predicted interaction outputs, the local decision units adjust the model output vector $\hat{y}$ so that when corresponding controls are applied to the real process, the real process output vector y_* is identical to the model output vector $\hat{y}$. The local optimisation problem (LOP_i) for each decision unit is,

$$
(LOP_i)\ \begin{cases} \text{Find } \hat{c}_i(s_i, \hat{y}) \text{ such that} \\ \hat{c}_i(s_i, \hat{y}) = \arg\min_{c_i \in \mathcal{C}_i} Q_i(c_i, H_i\hat{y}, \hat{y}) \\ \text{s.t.} \quad G_{ij}(c_i, H_i\hat{y}, \hat{y}_i) \in CUY_i, \quad j \in J_i \\ \hat{y}_i + s_i = F_i(c_i, H_i\hat{y}), \quad i \in \overline{1,N} \end{cases} \tag{5.4}
$$

$$
s_i(\hat{y}) = \{s_i \in \mathcal{Y}_i : \exists c_i, \hat{y}_i + s_i = F_i(c_i, H_i\hat{y}), G_{ij}(c_i, H_i\hat{y}, \hat{y}_i) \le 0, j \in J_i\}, \quad i \in \overline{1,N} \tag{5.5}
$$

An iterative updating scheme is required in the local decision problem to update the shift vector s,

$$
s^{n+1} = s^n + \varepsilon\,[I - F_u'(\hat{c}(s^n,\hat{y}),H\hat{y})H](\hat{y} - K_*(\hat{c}(s^n,\hat{y}))) \tag{5.6}
$$

$$
s_i^{n+1} = s_i^n + \varepsilon\,[(\hat{y}_i - y_{*i}^n) + (F_i)_{u_i}'(\hat{c}_i(s^n,\hat{y}),H\hat{y}) \sum_{j=1}^{N} H_{ij}(\hat{y}_j - y_{*j}^n)], \quad i \in \overline{1,N} \tag{5.7}
$$

Since $y_*^n \triangleq K_*(\hat{c},(s^n,\hat{y}))$ (5.8)

$$s_i^{n+1} = s_i^n + \varepsilon\,[(\hat{y}_i - y_{*i}^n) - (F_i)_{u_i}'(\hat{c}_i(s_i^n,\hat{y}),H_i\hat{y})(\hat{u}_i - u_{*i}^n)],$$
$$i \in \overline{1,N} \qquad (5.9)$$

The iterative algorithm of the local decision problem can be described in the following steps, given the initial interaction variables u_i, y_i and ε, and setting the program counter, n and shift vector $s_i^1 = 0$, goto step 3,

1) Measure y_{*i}^n, u_{*i}^n and if $y_{*i}^n = y_i^n$, local optimum solution is reached, otherwise proceed to next step;

2) Update the shift vector s using the scheme,

$$s_i^{n+1} = s_i^n + \varepsilon\,[(\hat{y}_i - y_{*i}^n) - P_i^n(\hat{u}_i - u_{*i}^n)]$$

where $P_i^n \triangleq (F_i)_{u_i}'(\hat{c}_i(s_i^n,\hat{y}),H_i\hat{y})$;

3) Evaluate $\hat{c}_i(s_i^{n+1}, \hat{y})$ according to (5.4) and apply to the subprocesses;

4) Wait until the real system transient has died down and repeat step 1 with n=n+1.

Once the infimal decision problem solution is reached, the solution set is then transmitted to the coordinator. The task of the coordinator, same as the open-loop, is to determine the interaction output to minimise the performance index Q. The coordinator problem (COP) is,

COP —

Find a coordination variable

$$\hat{y} = \arg\min_{y \in Y \cap Y_0} Q(\hat{c}(y),HK(\hat{c}(y)),K(\hat{c}(y))) \qquad (5.10)$$

where $y \triangleq \{y \in \mathcal{Y} : K(\hat{c}(y)) \in Y_{*_1} x \ldots x Y_{*_N}\}$

$Y_0 = \{y \in \mathcal{Y} : C_i(y) \neq \phi,\ i \in \overline{1,N}\}$

5.2.3 Interaction Balance Method with Global Feedback (IBMGF)

This closed-loop coordination method was first proposed by Findeisen (1974) and further investigated by Malinowski and Ruszczynski (1975), and Malinowski (1976) where feedback information from the real process is used by the coordinator. In applying the global feedback to the interaction balance method, the basic approach is to leave the local decision units the same as in the open-loop method to perform their optimisation tasks using the mathematical models. The local optimisation problem (LOP) is that, for a given coordination variable λ,

$$
\text{LOP} \left\{
\begin{array}{l}
\text{Find } \hat{c}_i(\lambda) \text{ and } \hat{u}_i(\lambda) \text{ such that} \\
\\
(c_i(\lambda), u_i(\lambda)) = \underset{(c_i, u_i) \in CU_i}{\arg\min} L_i(c_i, u_i, \lambda) \qquad (5.11) \\
\\
\text{where} \quad CU_i = \{(c_i, u_i) : G_i(c_i, u_i) \leq 0\} \\
\\
L_i(.) = Q_i(c_i, u_i) + \langle \lambda_i, u_i \rangle - \sum_{j=1}^{N} \langle \lambda_j, H_{ji} F_i(c_i, u_i) \rangle
\end{array}
\right.
$$

It is assumed that the solution $(\hat{c}_i(\lambda), \hat{u}_i(\lambda))$ of the local decision optimisation problem is unique for every $\lambda \in \Lambda$ where Λ is the solution set of the local decision problem. The model interaction inputs $\hat{u}_i$ are sent to the coordinator and the optimal controls $\hat{c}_i$ are then applied to the real subprocesses and the real interaction inputs $u_*(\lambda) = HK^*(\hat{c}(\lambda))$ are realised and transmitted to the coordinator.

The task of the coordinator is to determine the Lagrange multipliers λ to minimise the global imbalance R_* between the real interaction inputs u_*, and the model interaction inputs u, i.e.

$$R_*(\lambda) \triangleq \hat{u}(\lambda) - u_*(\lambda) = 0 \qquad (5.12)$$

Several iterative strategies to solve the coordinator optimisation problem have been proposed ranging from the simple 'direct minimisation' of the norm of $R_*(\lambda)$ to the more sophisticated Newton-like updating technique (Findeisen et al, 1980). In this simulation study, the former iterative scheme has been adopted to solve the coordinator optimisation problem (COP), i.e.,

$$\text{COP} \left\{ \begin{array}{l} \text{Find} \quad \tilde{\lambda} = (\tilde{\lambda}_1, \ldots, \tilde{\lambda}_N) \quad \text{such that,} \\ \\ \min_{\lambda \in \Lambda} \{ \| \hat{u}_i - u_{*i} \| \} , \qquad i \in \overline{1,N} \end{array} \right. \qquad (5.13)$$

5.2.4 Interaction Balance Method with Local Feedback (IBMLF)

This coordination strategy where feedback information from the real process is used by the local decision units was first proposed by Findeisen (1976) and subsequently investigated by Brdys and Michalak (1978), and Brdys and Ulanicki (1978). Each decision unit has the task of determining its local controls $c_i \in \mathcal{C}_i$ to minimise the local modified objective function $L_i(c_i, u_i, \lambda_i)$. The Lagrange multiplier, λ_i is specified by the coordinator. The interaction inputs u_i are determined based on the real interaction inputs, u_{*i} realised from the real process resulting from the previous application of all local controls c. For a given coordination variable λ and interaction variable u_i, the local optimisation problem (LOP_i) is,

$$
\text{LOP}_i \left\{ \begin{array}{l} \text{Find controls } \hat{c}_i(u_i,\lambda) \text{ such that} \\ \\ \hat{c}_i(u_i,\lambda) = \underset{c_i \in C_i(u_i)}{\arg\min}\, Q_i(c_i,u_i) + \langle \lambda_i, u_i \rangle - \sum_{j=1}^{N} \langle \lambda_j, H_{ji} F_i(c_i,u_i) \rangle \\ \\ \text{where } \; C_i(u_i) \triangleq \{ c_i \in \mathcal{C}_i : (c_i,u_i) \in CU_i \} \end{array} \right. \qquad (5.14)
$$

At each iteration of the LOP_i, the interaction inputs u_i is determined based on the feedback measurements from the real process. The existence of the LOP_i depends on the assumption of non-empty local problem feasible set U_0 where

$$
U_0 \triangleq \{ u \in \mathcal{U} : C(u) \neq \phi \} \qquad (5.15)
$$

$$
\text{where } \; C(u) = \bigtimes_{i=1}^{N} C_i(u_i)
$$

An iterative procedure to solve the local decision problem has been suggested which can be described in the follow steps, given the initial model interaction inputs u_i and the iteration counter k=0,

1) Solve the local optimisation problem (LOP_i),

$$
\hat{c}_i(u_i,\lambda) = \arg\min \{ Q_i(c_i,u_i^k) + \lambda^T (u_i^k - \sum_{j=1}^{N} H_{ji} F_i(c_i,u_i)) \}
$$

$$
\text{where } \; C_i(u_i) \triangleq \{ c_i \in \mathcal{C}_i : (c_i,u_i) \in CU_i \} \; ; \qquad (5.14a)
$$

2) Apply $\hat{c}_i$ to the real subprocess. Wait until all local decision units have reached this stage and the real system transient has died down;

3) Measure the real interaction input u_{*i} and if $u_{*i} = u_i^k$, local optimum solution is reached, otherwise proceed to next step;

4) Let $u_i^{k+1} = u_i^k + \varepsilon_i(u_{*i} - u_i^k)$, k=k+1, and repeat from step 1.

where ε_i is a diagonal gain matrix which may be tuned in an attempt to improve convergence. Generally, except in simple examples, suitable gain values of ε_i are difficult to obtain by theoretical derivation and trial and error methods are commonly used.

In order to ensure stable operation of the local decision iterative scheme, it is necessary to derive sufficient conditions for the convergence of such iterative scheme. Contraction mapping techniques have been used and it is found that (Brdys and Ulanicki, 1978) the Lipschitz's constant of the real interaction input mapping $HK_*(c(.,\lambda))$ must be less than unity. In many practical situations, it may be difficult to determine this constant. Hence, convergence of the local decision iterative scheme cannot be guaranteed.

Once the infimal decision problem solution $(c_b(\lambda), u_b(\lambda))$ is reached, it is then transmitted to the coordinator. The task of the coordinator is to determine the Lagrange multipliers λ so as to minimise the real performance index Q_*. The coordinator optimisation problem (COP) is,

COP

Find $\hat{\lambda} = (\hat{\lambda}_1, \ldots, \hat{\lambda}_N)$, such that

$$\hat{\lambda} = \arg \min_{\lambda \in \Lambda_b} Q_*(\lambda) \qquad (5.16)$$

where $Q_*(\lambda) \triangleq \sum_{i=1}^{N} Q_i(c_{bi}(\lambda), u_{bi}(\lambda))$, $i \in \overline{1,N}$

Λ_b is the set of all coordination variable values for which $u_b(\lambda)$ exists.

The above coordinator problem formulation is known as 'full coordination'. An alternative formulation of the coordinator problem known as 'partial coordination' is as follows,

$$\text{COP} \left\{ \begin{array}{l} \text{Find} \quad \lambda^0 = (\lambda_1^0, \ldots, \lambda_N^0) \text{ such that} \\ \\ u_b(\lambda) = HF(\hat{c}(u_b(\lambda), \lambda), u_b(\lambda)) \ , \quad \lambda \in \Lambda_b \end{array} \right. \tag{5.17}$$

In general, a solution obtained by using partial coordination is not better than that obtained by using full coordination. However, the solution obtained by partial coordination can be used as a starting point for full coordination since it is easier to find the solution using the former coordination strategy.

5.3 Synchronisation and Inter-process Communication

In the real-time implementation of the closed loop coordination methods using the distributed computer system, two local decision units are used in the infimal level. Parallel computation can be performed at the local decision level once coordination parameters have been received from the supremal level. Since first order transfer lags have been introduced within the simulated real process, the decision units need to wait sufficient time for the system transients to die down in order to obtain steady state measurements. The determination of a minimum waiting time for different transfer lag time constants has been carried out off-line using the simulation software package 'ISIS' available on a Prime 550 mini-computer.

Using the distributed hierarchical optimising control approach, there exists information exchange between

(a) coordinator and local decision units for coordination methods with global feedback or local feedback;

(b) each local decision unit for coordination methods with local feedback.

Since parallel processing is performed in the infimal level and each local decision unit optimisation iteration finishes at a different time interval, synchronisation between the two levels is required. Various synchronisation mechanisms for distributed computer systems have been studied. Since the hierarchical computer structure consists of only three interlinked computers, therefore, simple synchronisation mechanisms such as elapsed time and semaphore techniques are used to synchronise the decision units so that controls can be sent to the real-subprocess simulataneously.

(1) Elapsed Time

Estimate the time required for each decision units to finish one optimisation iteration. Each decision units then waits until the slowest unit to complete its task before sending the controls to the real process and taking measurements at steady state.

(2) Semaphore

After each optimisation computation, each decision unit sends a completion flag to the coordinator and waits. A task of the coordinator is to check that all the completion flags from the decision units have been received before transmitting a start flag to each decision unit, all of which then send controls simulataneously to the real process and take measurements at steady state. The start flag is then reset for the next data set.

The advantage of the first proposed synchronisation scheme is that total decentralisation at the local decision unit level can be obtained. However, it is in practice very

difficult to determine the waiting time for each decision units because the computation time required at each optimisation iteration will often vary. This may affect the stability and convergence at the infimal level. Therefore only the second method has been adopted to synchronise the decision units although this requires extra information exchange between the coordinator and the local decision units. This synchronisation scheme has been shown to work efficiently during real-time control of the simulated real process under different coordination methods with feedback.

Using the local feedback scheme, a further synchronisation problem arises because each decision unit converges to its final solution over a different time horizon. The semaphore method is again used to synchronise the coordinator and the local decision units because the existing theory suggests that the coordinator should adjust its parameters only when all the decision units have converged to their solution.

An investigation had been carried out where the decision units were not synchronised before applying the controls to the real process and it was observed that, as a result, the decision units became unstable. Although decision units could be stabilised by reducing their iteration loop gain parameters, this decreased the convergence rate of the local optimisation problem. In this particular simulation study, the synchronised scheme required far fewer iterations in the local decision level, and thus the total iterations required for the global optimisation was reduced.

5.4 Asynchronous Iteration for Closed-loop Hierarchical Control and Optimisation of Interconnected Systems

According to the hierarchical control and optimisation theory developed for large scale systems operating at steady state condition, the theory requires synchronisation

in the infimal level of the hierarchical structure. i.e. the local decision units have to wait until the slowest unit finishes its optimisation task before the optimal controls are applied to the real subprocesses. Hence, some obvious implementation disadvantages arise using the synchronous hierarchical algorithms such as algorithm initialisation and iteration synchronisation protocol are needed. Furthermore, the speed of computation in the infimal level is restricted to that of the slowest decision unit. These limitations can be reduced by developing an algorithm to minimise the degree of synchronisation and inter-process communication. Various asynchronous algorithms suitable for distributed computation have been proposed by Baudet (1978), Bertsekas (1983) and Tsitsiklis (1986). However, due to the lack of memory and computing power available in the I-MICs, the proposed algorithms cannot be implementated within the I-MICs.

Stability is the main concern in control system design for interconnected process when the local decision units are operating asynchronously in the infimal level within the hierarchical structure. Instability occurs in the local decision level due to the following two causes. Firstly, the updation of the interaction variables $\underline{u}_i(k+1)$ depends on the values of $\underline{u}_{i*}(k)$ which is a function of all controls $\underline{c}_j$, $j \neq i$. Hence, if the local decision units are not synchronised before sending the controls to the real process and taking measurements, the value obtained will not be $\underline{u}_{i*}(k)$, but rather it will be $\underline{u}_{i*}(k-T)$, where T is the time difference between the iteration time required by each of the two decision units. This time difference can be viewed as *delay* which occur frequently in process control problems. The presence of *delay* is undesirable in feedback control since the control action is based on the delayed information and the resulting phase lag tends to make the system less stable and hence more difficult to achieve satisfactory control. Furthermore, time *delay* in hierarchical interconnected systems make control system design more difficult since the design approaches are not

applicable to time delayed systems. Secondly, because first-order time lags are introduced to the controls and the interconnections, it is possible that just before taking the steady state measurement from the real subprocess i, $\underline{u}_{i*}$ is disturbed by controls $\underline{c}_j$, $j \neq i$. Hence measurement $\underline{u}_{*i}$ will not be a steady state value which again will affect the stability of the local iteration.

'Asynchronous iteration' in the infimal level means that computation and communication is performed at various decision units completely independent of the progress in other units.. In studying the effect of asynchronous iteration upon the convergence and stability of the overall system, we only considered the coordination methods with local feedback. i.e. IBMLF and IPMLF. Constrained Simplex algorithm and the 'Active Set Method' were employed to solve the coordinator problem and the local decision problem respectively. The asynchronous local decision iteration algorithm was the same as those of the synchronous one except the application of controls to the simulated process were not synchronised. An investigation on the direct application of such asynchronous iteration algorithm to on-line hierarchical control with local feedback had been performed. The performance of this algorithm operating in the real-time environment will be described in the following chapter.

5.5 Summary

The on-line coordination methods suitable for closed-loop hierarchical control and optimisation of large scale interconnected systems were described. This included the Interaction Balance Method (IBM) and the Interaction Prediction Method (IPM) incorperated with global or local feedback. For each closed-loop coordination schemes, the optimisation algorithms for the coordinator and the local decision units were outlined.

In the real-time implementation for closed-loop hierarchical control of interconnected process, the problems encountered such as synchronisation and inter-process communication had been investigated. The method of semaphore was chosen to synchronise the local decision units. Since first order transfer lags have been introduced within the simulated process, off-line simulation using the software package ISIS was performed in order to determine the minimum waiting time for steady state measurements.

Problems associated with asynchronous iteration for closed-loop hierarchical control and optimisation of interconnected systems were briefly described. Finally, the application of asynchronous local decision iteration algorithm suitable for closed-loop hierarchical control and optimisation has been suggested.

CHAPTER 6 SOFTWARE DEVELOPMENT

6.1 Introduction

In this chapter, the software development for the hierarchical structure is discussed. It is a two-level structure which consists of a coordinator in the supremal level, two local decision units in the infimal level and a simulated real process. At each level, the software is essentially composed of two sections, the objective section and the information exchange section. The function of the objective section is to perform the assigned task, e.g. to optimise a constrained function (performance index) in order to generate optimal controller set points. The function of the information exchange section is to transmit the solution of the assigned task (objective section) to, and receive information from, other levels within the hierarchy for data processing and control purposes.

The software development for the coordinator, local decision units and the simulated real process will be outlined in section 6.2, 6.3, 6.4 respectively.

6.2 Coordinator (LSI-11/23) Software

The coordinator software is resident in the LSI-11/23 computer. The functions of this software are to transfer data to and from the local decision units, to determine the optimal coordination variables, to produce a hard copy output, to plot the controls and interaction variables on the Tektronix 4010 graphics terminal and to display current data from the coordinator and the local decision units in a mimic diagram on the Intecolor 8001G graphics terminal. Since the LSI-11/23 computer is operating under the TSX-Plus time-shared operating system, a multi-user operating system, on-line display on the graphics terminal

has to be suppressed for the benefit of other system users. Therefore, output data obtained from the on-line simulation were logged on the floopy disk and the controls and interaction variables were plotted off-line.

The coordinator software consists of two sections, the objective section and the information exchange section. The latter section is concerned with time-critical data transfer with the local decision units (I-MICs). Since each I-MIC deals with time-critical DDC functions, whereas the LSI-11/23 deals with less time-critical user interface functions, each data transfer must be initiated by the I-MIC. A foreground/background approach has been adopted in developing the coordinator software. The foreground program deals with time-critical functions, i.e. data transfer with I-MICs, and is assigned with a higher execution priority. Assigned with a lower execution priority, the background program optimises the constrained coordinator decision problem and logs the output data for future use.

The following subsections describe the foreground and background routines that constitute the coordinator software.

6.2.1 Program Units

The program units required to run the foreground coordinator software are as follows:

LKSET2	Subroutine to initialise two I-MIC links.
RK	Asynchronous completion routine used to maintain the main buffers for data received over the I-MIC links.
LINKXD	Data acquisition subroutine for operating two I-MIC links.

The program units required to run the background coordinator software are as follows:

IPMGFB,IPMLFB, IBMGFB,IBMLFB, IBMLFA,etc	Main (optimisation) programs used to determine the optimal coordination variables under different coordination schemes.
SYNCH	Subroutine used to synchronise the data transfer between foreground/background program.
GET2,IBGGET, IPGGET,IPLGET, etc	Subroutine used to transfer data from foreground program to the background program and determine the performance indices and the real interaction variables.
IBMF23,IBMGFF, IPMGFF,IPMLFF, etc	Subroutine used to communicate asynchronisely with I-MICs through the links.
SDIC	Subroutine to output data for Intecolor display.
ICSUB	Subroutine to send data from background program to the Intecolor.
TIDY2	Subroutine used to disable the receipt and the transmission of interrupts set by link operation routines.
TIME	Subroutine to print the current time.
DATEO	Subroutine to print the current date.
CON1	Subroutine used to check for constraint violation when coordinated by Interaction Prediction Method.

6.2.1.1 Foreground Routines

The routines that form the foreground software are subroutines LKSET2, RK and LINKXD. The aim of the foreground routines is to transfer data between the coordinator (LSI-11/23) and the two local decision units (I-MICs). These routines were originally written by Dr.W.J. Hill in MACRO 11 assembly language and was subsequently modified by Dr. I.A. Stevenson to ensure reliable data transfer. Furthermore, the software was rewritten entirely in FORTRAN and is thus more easily understood than the original MACRO routines. Detailed software description of these routines can be found in Stevenson (1982,1984,1985). A full listing of the foreground routines is given in appendix B1.

6.2.1.2 Background Routines

According to their functional nature, the background routines can be classified into three groups:

Optimisation routines - main program and subroutine CON1;

Data transfer routines - subroutines SYNCH, GET2, IBMF23 and TIDY2;

Schematic output routines - subroutines SDIC, ICSUB.

The program units used to run the background software are the same for different coordination method except the main (optimisation) program where two different routines are employed. Therefore, the listing of the background routines is made up of the main problem - constrained Simplex (IBMLF, IPMGF, IPMLF) algorithm or the steepest descent algorithm (IBMGF) and the other supporting routines which can be found in appendix B2.

Since most routines are quite simple and self explanatory (see appendix B2), only the main program and the subroutine IBMF23 were chosen to be described briefly in the following subsection.

6.2.1.2.1 Main Program

The main program has the task to solve the coordinator optimisation problem. An optimisation algorithm is used to generate optimal coordination variables for the local decision problems. The optimisation algorithm employed to solve the coordinator problem depends on the coordination method used. For IBMGF, steepest descent algorithm is used. Constrained simplex method has been employed for IBMLF, IPMGF and IPMLF. The selection of appropriate optimisation algorithm for coordinator problem depends upon the availability of the gradient vector of the coordinator performance function. In general, the steepest descent algorithm gives a faster rate of convergence than the constrained Simplex algorithm.

The first part of the main program is the initialisation. Data files are created so that controls and interaction variables during on-line program execution can be logged on the floppy disk for future use. It also initialises the Intecolor display and the I-MIC links by calling the subroutine LKSET2. System routines TRMASC and IPOKE are called to enable an interrupt in case of system shut down through the console command. All the variables required for the optimisation algorithm are initialised.

The second part of the main program is the optimisation alogrithm. For the steepest descent algorithm, the iterative procedure can be summarised in the following steps:

1) Start with the initial estimate X_i, and set the counter $i = 1$;

2) Find the search direction $S_i = - \nabla f_i$;

3) Find λ_i^* to minimise $f(X_i + \lambda_i S_i)$;

4) Set $X_{i+1} = X_i + \lambda_i^* S_i$;

5) Check X_{i+1}, if $(X_{i+1} - X_i)$ is within the prescribed tolerance, when X_{i+1} is optimal, stop; otherwise set i=i+1 and goto step 2).

Using the constrained Simplex algorithm, the following steps are performed:

1) Estimate the first Simplex vertice and its function value;

2) Generate the rest of Simplex vertices and their function values;

3) Perform tests and re-order function values;

4) Check the Simplex size, if the Simplex size is less than the prescribed tolerence, stop. Otherwise proceed to next step;

5) Perform the action of reflection / reduction / contraction / extension, which one to be performed depends on the function value of the vertices;

6) Goto step 3).

The third part is the solution output section where the optimal solution set is logged on the output data file. Once the optimal solution of the coordinator has been reached, stopping signals are sent to each I-MIC to inform them not to generate further interrupts. Then subroutine TIDY2 is called to disable the interrupts that are set in

response to the signals from I-MICs.

6.2.1.2.2 Subroutine IBMF23

This subroutine is used to communicate asynchronously with the I-MICs through the links. When a data transfer initiated by the I-MIC takes place, the foreground programs LINKXD and RK are called. The data temporarily stored in the buffers of the subroutine RK in the foreground is then transferred to the buffers of the subroutine IBMF23 in the background which is then used by the main program. Successful data transfer between the foreground and the background program depends on the status control of various flags and bytes, e.g. TXFREE, TFLAG, DATAR, DATAX etc.

6.3 Local Decision Unit (I-MIC) Software

Foreground/background approach is employed again in developing the I-MIC software. The foreground program deals with the time-critical data transfer with the coordinator (LSI-11/23) and the simulated real process (TR48). The background program solves the local decision optimisation problem in order to generate optimal controls. The interpretive language CONTROL BASIC (CB) is used to program the background program. The foreground program is programmed in 8085 machine code via the I-MIC monitor since CB is not fast enough in time-critical data transfer applications. The machine code subroutine occupies memory locations 2400(HEX) - 2453(HEX) inclusive (on the F030 processor board). A CB subroutine starting at line number 30000 and ending on line number 32030 is resident in the background program. The I-MICs communicate with the LSI-11/23 via calls to the CB subroutine at line number 30000.

When data transfer is required, the foreground program is activiated by calling the CB instruction "SCH" in the

background program. Once the foreground program finishes its assigned task, control returns to the background program.

The I-MIC background program determines the optimal controls to be applied to the simulated interconnected process. Since the I-MIC uses only integer numbers in the range of -32767 to 32768, it is necessary to scale variables used in the program in order to maintain a prescribed accuracy. However, this may cause difficulties with numeric overflows. After some trial tests, a scaling factor of 1000 is chosen to be used in the I-MIC program thereafter. Basically, there are three sections in the I-MIC software, namely, the initialisation, optimisation and data transfer. However, if local feedback coordination is used, an extra section called "parameter estimation" is required.

The CB I-MIC program can be summarised in the following sections:

1. Initialisation;

2. Optimisation;

3. Synchronisation (with coordinator);

4. Data transfer to and from the simulated interconnected process;

5. Checking convergence of the local decision problem;

6. Data transfer to and from the coordinator.

The program starts with initialisation. All arrays and program counters are initialised. It also produces a program log for future reference.

The second section of the program is the optimisation routine. Based on a steady state mathematical model, optimal controls are computed which will subsequently be applied to the simulated real processes. Due to the lack of programming memory, mathematical functions and subroutines available in I-MIC, the optimisation problem has to be solved analytically using the theory of extrema and the idea of 'active set method'. A set of feasible controls and the corresponding performance index are generated using this analytical approach. Optimal controls are those associated with the minimum performance index. Once the optimal controls and the corresponding performance index have been determined, the model based interaction variables are computed. These model based interaction variables and optimal controls are useful parameters for local and/or coordinator optimisation problem. Using the 'active set method', the feasible solution set of the local optimisation problem for various closed-loop coordination method has been generated which can be found in appendix A.

The third section contains the data transfer or interfacing routine with the coordinator and the simulated real process. For data transfer between the I-MIC and the analogue computer, the following instructions are performed:

Data transfer from I-MIC to analogue computer:

APO.(H.3E04,H(5))	: Write data stored in H(5) in location 3E04.
APO.(H.3E06,I(5))	: Write data stored in I(5) in location 3E06.
PO.(H.3200,0)	: Send the data stored in location 3E04 and 3E06 to analogue computer using the ADC/DAC module G226/G266.

W.(500)	: Wait 500 machine units for the analogue system transient to die down in order to obtain the steady state real interaction variables.

Data transfer from analogue computer to I-MIC:

PO.(H.3E20,0); W.(5)	: Initialise the location 3E20 by writing a zero to it and wait for 5 machine units to ensure error free data transfer.
B(0)=APE.(H.3E20)	: Read the data stored in the location 3E20 and assigned as B(0).
PO.(H.3E22,0); W.(5)	: Initialise the location 3E22 by writing a zero to it and wait for 5 machine units to ensure error free data transfer.
B(1)=APE.(H.3E22)	: Read the data stored in the location 3E22 and assigned as B(1).

The model based optimal controls are transmitted to the analogue computer and steady state real interaction variables are realised from the analogue computer using the interface card G226/G266.

When optimal solution of the local decision problem has been reached, data are transmitted to the coordinator (LSI-11/23) using the I-MIC - LSI link routine. Data are stored in the array Y of the I-MIC routine prior to the data transfer. The contents of the array are output bytes using the following instructions:-

```
30070  F.J=1 to Y(0)     : Y(0) = byte count

30080  U=Y(J)

30090  GOS.32000

30100  U.H.2400(U)       : reconstitutes the bytes received
                           from the LSI-11/23 into words by
                           transposing the low and high
                           bytes of the data word in between
                           receiving operations and after.

30110  GOS.32000

  .       .
  .       .
  .       .

30120  N.J
```

```
32000  U.H.240A()        : disables all the processor
                           interrupts to the I-MIC, tidies
                           up the stack and returns.

32010  U.H.C8(U)         : a monitor routine which outputs a
                           data byte (low byte of the
                           argument) to the serial output
                           line.

24015  U.H.2412(K)       : ensures communication routine
                           handle all the interrupts with
                           minimum delay by waiting to
                           receive a byte (zero) after
                           transmitting one.

32020  U.H.240E()        : reenables the processor
                           interrupts, tidies up the stack
                           and returns.

32030  R.
```

For coordination using local feedback scheme, a parameter estimation section is included. Based on the model and the real interaction variables, an updating formula has been established which generates improved interaction variables for stable and fast converging iterative local decision optimisation problem.

6.4 Simulated Interconnected Process (TR48)

An analogue computer was used to simulate two interconnected sub-processes under direct control from local decision units. The block diagram which relates the input/output of the interconnected sub-processes is shown in fig.6.1.

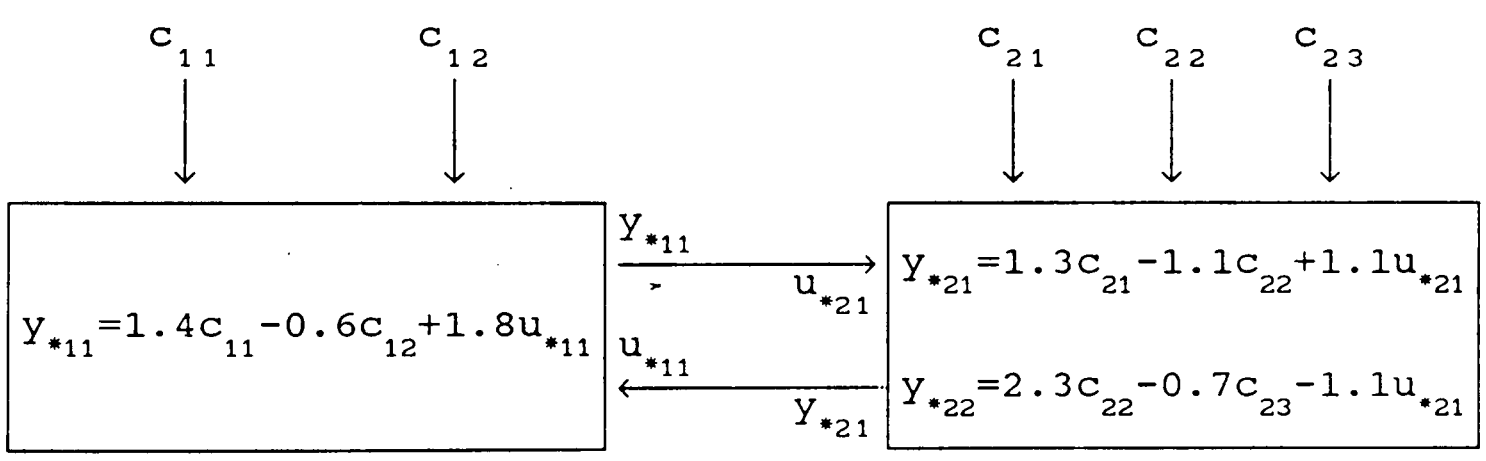


Fig.6.1 Schematic diagram of the interconnected sub-processes

First order time constants were introduced to the interaction inputs and the controls within the analogue computer to simulate the effect of transfer lags upon the stability and convergence of local decision optimisation problems. The response of the interaction inputs due to these transfer lags was studied initially using the software package called "ISIS". It was then implemented to the analogue computer.

Using the G226 DAC module, optimal control inputs from the local decision units were sent to the analogue computer

and the steady state real interaction inputs from the analogue computer were realised and transmitted back to the coordinator or the local decision units via the G266 ADC module depending on the feedback mechanism used.

In order to improve the accuracy of the simulation result, amplitude scaling was required. Based on the open loop solution from the off-line simulation on the Prime 550 computer, the optimal controls from the local decision units were sent to the analogue computer. Tests had been carried out to determine the optimal set of scaling factors on the controls so that reasonable accurate result can be obtained. The optimal settings were found after some trial runs by setting a set of controls from the decision units to the analogue computer. Real interaction inputs were measured, which were then compared with the model based real interaction inputs that were computed based on the control sets. It was found that they were very close with a maximum deviation of unity of the last digit. This suggested that the settings in the analogue computer were accurately representing the mathematical model of the interconnected sub-processes. For all the subsequent test runs for different coordination methods, the analogue computer setting was checked in the initialization procedure. A set of controls were transmitted to the analogue computer and the real interaction inputs were realised which were then compared with the model based solution so as to ensure the simulation run was working with a reliable and consistant analogue computer setting.

The main problem encountered when using the TR48 analogue computer for simulation was the voltage drift. The computer had to be powered up for more than ten minutes prior to the simulation run in order to warm up the electronic circuits of the analogue computer and to stabilise the amplifiers.

6.5 Summary

The software development of the two-level hierarchical control system has been described. This includes a brief description of the routines that had been used for the coordinator, the local decision units and the analogue computer.

For the coordinator software, the optimisation algorithms (the constrained Simplex algorithm and the Steepest Descent algorithm), the data transfer routines and the schematic output routines were implementated, together with the foreground routines, to solve the coordinator optimisation problem. For the local decision unit software, the optimisation problem was solved analytically using the 'active set method' for different coordination strategies. Communication between the I-MICs and the analogue computer are via the ADC/DAC modules. Problem encountered when using the analogue computer for simulation was noted and remedial measure was taken.

Having described the on-line coordination methods, the hardware and the software developed for on-line hierarchical control in previous chapters, investigation of the applicability of the software, the theory developed for on-line control and optimisation of a simulated interconnected two-level hierarchical structure is performed and simulation results based on different closed-loop coordination method will be given in the next chapter.

CHAPTER 7 SIMULATION RESULTS AND DISCUSSION

7.1 Introduction

Using the two-level hierarchical computer structure, simulation tests had been performed to investigate the closed-loop hierarchical control and optimisation of a simulated interconnected process. In order to ensure stable and convergent coordinator and local decision problems, the model based optimal controls generated by the decision units had to be synchronised before applying to the simulated real subprocesses. For synchronised local decision iteration, all the closed-loop coordination methods (i.e. IBMGF, IBMLF, IPMGF, IPMLF) had been implemented and simulation tests were performed.

Tests had also been carried out to investigate the effects on system stability and convergence of local decision optimisation problem when local decision units were operating asynchronously. Only the local feedback coordination schemes, i.e. IBMLF and IPMLF had been implemented and simulated by the hierarchical computer system.

Two optimisation methods were employed to solve the coordinator optimisation problem for different coordination strategies. Direct search method - the constrained Simplex algorithm was used for IBMLF, IPMGF, IPMLF. Gradient method - the steepest descent algorithm was employed to solve the IBMGF coordinator optimisation problem. Owing to the lack of memory and software support from the I-MICs, analytical method based on the 'Active Set Method' was used to solve the local decision optimisation problem for different on-line coordination method.

Simulation test results based on synchronous and asynchronous local decision iteration will be presented in

the following section 7.2 and 7.3 respectively. The graphical output of the simulation results will be shown in section 7.5.

7.2 Synchronous Local Decision Iteration

Based on the hierarchical control and optimisation theory (Mesarvoic,1970; Findeisen,1980), the optimal controls generated by the local decision units should be sent to the interconnected subprocesses simultaneously. Since each local decision task would finish at different time horizon, synchronisation of local decision units was required in order that optimal controls could be applied to the subprocesses concurrently. Method of 'semaphore' was chosen to synchronise the local decision units in the infimal level for direct control of the simulated interconnected subprocesses. The chosen synchronisation mechanism provided a reliable means of data transfer in the simulation test runs.

7.2.1 Interaction Prediction Method with Global Feedback

For this coordination strategy, feedback information was used by the coordinator to update the coordination variables for optimal overall system performance. The coordinator updated the coordination variables using the 'synchronised' measurements from the simulated real process. Using the open-loop solution as the starting point, i.e. $\underline{V}$ = (1.171; 0.965), the coordinator optimisation problem converged to its optimal solution

$$\underline{V} = (1.137,\ 1.005)$$
$$\underline{c} = (-0.676, 0.197;\ 0.868, 1.0, -1.0)$$
$$\underline{y}_* = (1.033;\ 1.166, 1.864)$$
$$Q_* = 5.9326$$
$$\Delta Q = 0.110\%$$

after 32 iterations. The coordinator took 29.7 sec/iteration depending on the number of users connected to the host computer during the test run. Simulation results are shown in fig.7.2.1.1.

With different starting point, e.g. $\underline{V}$ = (0.0; 0.0), it took 78 function calls before the coordinator converged to its solution

$$\underline{V} = (1.112,\ 1.006)$$
$$\underline{c} = (-0.703, 0.197;\ 0.894, 1.0, -1.0)$$
$$\underline{y}_* = (1.014;\ 1.176, 1.885)$$
$$Q_* = 5.9345$$

The coordinator took 26.3 sec per iteration. Simulation results are shown in fig.7.2.1.2.

Examining the results shown in fig.7.2.1.2, the coordinator required more function calls before reaching its solution when compared with using open-loop solution as the starting point. However, it should be noted that the '$\underline{V}$', '$\underline{c}$' and '$\underline{y}_*$' were quite stationary after 40 coordinator function calls.

Tests had also been performed by introducing disturbances to the interaction inputs and/or the controls of the simulated subprocesses by varying the potentiometer settings within the analogue computer. Three tests have been carried out to model the disturbances in 'c's and 'u's, and local decision unit failures which are summarised as follows:

a) Modelling disturbances in controls

Starting with $\underline{V}$ = (0.0; 0.0), the system waited until the steady state solution had been reached before setting the control c_{21} off for one iteration using the TR48 function switch. Thus a big disturbance was resulted and the performance index shot up from 5.9389 to 20.0376.

However, the system returned to its optimal performance 5.9372 after 15 iterations! The final solution was:

$$\underline{V} = (1.124;\ 1.009)$$
$$\underline{c} = (-0.699, 0.195;\ 0.885, 1.0, -1.0)$$
$$\underline{y}_* = (1.03;\ 1.181, 1.867)$$
$$Q_* = 5.9360$$

The results are shown in fig.7.2.1.3.

b) Modelling LDUs failure

Starting with open loop solution, i.e. $\underline{V}$ = (1.171; 0.965), the system reached the steady state condition after 30 iterations. Investigation on system stability when subjected to local decision unit (LDU) failures was performed. The LDU failures were simulated by switching off the controls that applied to the TR48. e.g. controls c_{11} and c_{12} were switched off to simulate the failure of LDU1. Three LDU failure conditions were being tested, i.e.

i) LDU1 failure

By disconnecting the controls c_{11} and c_{12} to the TR48 for 3 iterations (from iteration 43 to 45), the Q_* shot up from 5.9393 to 11.1753 and returned to 5.9327 after 16 iterations.

ii) LDU2 failure

Controls c_{21} and c_{22} were off for 4 iterations (from iteration 73 to 77). Q_* shot up to 6.0055 and returned to 5.9360 after 9 iterations.

iii) Both LDU failure

All controls were off for 6 iterations (from iteration 100 to 106). The Q_* rose to 11.5265 and then to 17.2509 and returned to 5.9370 after 17 iterations.

The final solution was:

$\underline{V}$ = (1.131; 1.002)
$\underline{c}$ = (-0.674,0.199; 0.871,1.0,-1.0)
$\underline{y}_*$ = (1.029; 1.163,1.868)
Q_* = 5.9357

The results are shown in fig.7.2.1.4.

c) Modelling interconnections failure

Starting with the open-loop solution and the system waited until the steady state solution before setting the interactions off for 4 iterations using the TR48 function switch. The system was disturbed and the performance index shot up from 5.9338 to 10.6606. However, the system returned to its optimal performance 5.9367 after 15 iterations with the final solution:

$\underline{V}$ = (1.127; 1.016)
$\underline{c}$ = (-0.714,0.191; 0.889,1.0,-1.0)
$\underline{y}_*$ = (1.039 ; 1.196,1.857)
Q_* = 5.9372

The results are shown in fig.7.2.1.5.

Referring to the figures 7.2.1.3, 7.2.1.4 and 7.2.1.5 when disturbances were introduced to the system operating at its optimal steady state operating condition, the overall performance index increased. This was because the performance index of each local decision unit was a function of the controls and interaction inputs which in turn affected the overall (coordinator) performance index. After few iterations, the coordinator optimisation problem converged to a new 'disturbed' performance index. However, if the disturbances were removed, this 'disturbed' performance index would soon return to the optimal one after few iterations.

The above simulation results signified the robustness property of this coordination scheme when the controls and/or interaction variables were corrupted with noise. The system response to disturbances was stable and the system optimisation problem converged to a new steady state value rather fast.

7.2.2 Interaction Prediction Method with Local Feedback

Local decision units used the feedback information for this coordination strategy. For each set of coordination variables, there should exist a unique solution for each local decision problem. The task of each decision unit was to determine the optimal controls and interaction variables from a given set of coordination variables using the feedback information from the simulated real process.

Investigation of the optimal gain settings for stable and rapid converging local decision optimisation problem had been performed. Theoretical study on the stability of local decision optimisation problem using local feedback scheme was carried out using the 'Active Set Method'. A stable gain space was obtained. After some trial runs within the stable gain space, optimal gain settings for fast converging local decision optimisation problems were obtained as follows:

$$K = \begin{bmatrix} 0.8 & 0 \\ 0 & -0.8 \end{bmatrix}$$

Using the open-loop solution set, i.e. $\underline{V}$ = (1.171; 0.965) as the first estimate for the coordination parameters and with shift vector $\underline{s}$ = (0; 0), simulation test showed that the coordinator reached its solution

after 43 coordinator function calls while the LDU requires a total of 116 iterations. Time required for this simulation run took 53.6 minutes. The final solution was found as follows:

$$\underline{s} = (0.393;\ -0.319)$$
$$\underline{V} = (1.034;\ 1.085)$$
$$\underline{c} = (-0.594, 0.149;\ 0.808, 1.0, -0.941)$$
$$\underline{y}_* = (1.034;\ 1.086, 1.821)$$
$$Q_* = 5.9747$$
$$\Delta Q_* = 0.820\%$$

At the beginning of the simulation, each local decision optimisation problem required about seven iterations before converged to its solution for a given set of coordination variables. Simulation results are shown in fig. 7.2.2.1.

With $\underline{s} = (0;\ 0)$, tests had also been performed using different starting point, e.g. $\underline{V} = (1.118;\ 1.01)$ - the optimal solution of IPMGF, similar results were obtained as in fig.7.2.2.1.

7.2.3 Interaction Balance Method with Global Feedback

The steepest descent algorithm was used to solve the coordinator optimisation problem. Starting with $\underline{\lambda} = (0;0)$, the coordinator took 9 iterations to reach the open-loop solution, i.e. $\underline{\lambda} = (-2.537;\ -0.739)$. The coordinator required a total of 15 function calls before converged to its final solution

$$\underline{\lambda} = (-2.994;\ -0.387)$$
$$\underline{c} = (-0.609, 0.147;\ 0.824, 1.0, -1.0)$$
$$\underline{y}_* = (1.015;\ 1.086, 1.883)$$
$$Q_* = 5.9891$$
$$\Delta Q_* = 0.986\%$$

in 6.73 minutes. Simulation results are shown in fig.7.2.3.1.

In studying the effect of disturbances on the controls and interaction variables for this coordination method, several tests had been performed whereby the disturbances were simulated by varying the setting of the poteniometers within the analogue computer.

a) Modelling disturbances in controls

Halved c_{11} for 11 iterations (from iteration 23 to 33), Q_* rose from 5.988 - 7.88 - 7.241. Reset c_{11} to its full value at iteration 34, Q_* returned to 5.989 after 8 iteration (iteration 41).

Halved c_{12} at iteration 45; Q_* rose from 5.989 - 6.133 - 6.159 (iteration 50).

Simulation results are shown in fig.7.2.3.2.

b) Modelling disturbances in LDUs

Halved controls c_{11} and c_{12} at iteration 10, Q_* rose from 5.987 - 8.144 - 7.529 - 7.538 (new optimal performance at iteration 19). At iteration 26, c_{11} and c_{12} were reset to their full values, Q_* converged back to 5.988 after 6 iteration (iteration 32).

Halved controls c_{21} and c_{22} at iteration 41, Q_* rose from 5.985 - 6.076 - 6.19 (iteration 56).

Simulation results are shown in fig.7.2.3.3.

c) Modelling the interconnection failure

Set interconnection input u_{*11} off using the potentiometer coefficient in the TR48, the system became unstable. Investigation upon the effect of varying the

potentiometer coefficient related to u_{*11} upon the stability and convergency had been carried out. By varying the potentiometer coefficient (alpha) from 0.5 to 0.95, five tests had been performed and the following results were obtained:

No.	Alpha	Stability
1	0.5	Unstable
2	0.75	Oscillates, returned to solution if alpha=1
3	0.875	Same as 2
4	0.95	Converged to new optimal solution
5	0.91	Same as 2

Basing on the results obtained, the system coordinated by IBMGF was very sensitive to the disturbances imposed on the controls and especially on their interaction inputs. The simulation test showed that the system would become unstable if the interaction varaible was reduced by more than 5%.

Simulation results for alpha = 0.75 and 0.95 are shown in fig.7.2.3.4 and fig.7.2.3.5 respectively.

7.2.4 Interaction Balance Method with Local Feedback

For this coordination strategy, the study of stability and rate of convergence of the local decision optimisation problems were required. Investigation of the optimal gain settings for stable and rapid converging local decision optimisation problem had been performed. Theoretical study on the stability of local decision optimisation problem using local feedback was carried out using the 'Active Set Method'. A stable gain space was found. By the method of trial and error within the stable gain space, the optimal gain settings for fast converging local decision optimisation problems were obtained as

follows:

$$K = \begin{bmatrix} 0.3 & 0 \\ 0 & -2.5 \end{bmatrix}$$

Starting with the open-loop solution, i.e. $\underline{\lambda}$ = (-2.546; -0.744) and $\underline{u}$ = (0.965; 1.169), the coordinator took 88.9 minutes for 33 function calls before converging to its optimal solution

$\underline{\lambda}$.= (-2.731, -0.538)
$\underline{c}$ = (-0.708,0.12; 0.891,1.0,-0.99)
$\underline{y}_*$ = (0.978; 1.134,1.917)
$\underline{Q}_*$ = 5.9624
ΔQ_* = 0.613%

Simulation results are shown in fig.7.2.4.1.

Initially, the LDUs took about 8 iterations before converging to the solution for a given set of coordination variables. It took much less iterations as the coordinator approach its solution, typically 2 to 3 or less iterations. However, sometimes the LDUs might take more iterations before converging to the solution. This was due to the high gain, i.e. $\underline{K}$ = (0.3; -2.5) used in the LDU parameter updating equation which perturbed the system dynamics.

A test had also been performed by varying the time delay introduced in the analogue computer TR48, e.g. T_{ij} = 1 sec. similar simulation results were obtained with Q_* = 5.9664. This suggested the delays in the TR48 had a little effect on stability and system performance when the local decision units were operating synchronously.

7.3 Asynchronous Local Decision Iteration

Simulation tests had been carried out whereby the local decision units were not synchronised before sending the model based optimal controller set points to the analogue computer. This asynchronous operation could cause overall system instability.

7.3.1 Interaction Prediction Method with Local Feedback

With the starting point $\underline{V}$ = (1.171; 0.965) and $\underline{s}$ = (0; 0), the coordinator converged to the solution after 65 function calls with

$\underline{s}$ = (0.394; -0.321)
$\underline{V}$ = (1.035; 1.085)
$\underline{c}$ = (-0.592,0.149; 0.805,1.0,-0.941)
$\underline{y}_*$ = (1.037; 1.086,1.818)
Q_* = 5.9755
ΔQ_* = 0.833%

The local decision units, LDU1 and LDU2 took a total of 176 and 170 iterations respectively. Initially, each LDU required about 9 iterations before converging to the solution. It should be noted that Q_* = 5.9754 at iteration 13.

Simulation results are shown in fig.7.3.1.1.

Using $\underline{V}$ = (1.171; 0.965) and $\underline{s}$ = (0; 0) as the starting point, two tests had been performed by introducing a delay of 5 seconds in either one of the decision units prior to the application of the controls to the simulated subprocesses. With LDU1 delayed by 5 seconds, the coordinator, LDU1 and LDU2 took 32, 94 and 111 iterations respectively before converging to the solution:

$\underline{s}$ = (0.394; -0.324)

$\underline{V}$ = (1.046; 1.092)

$\underline{c}$ = (-0.597,0.145; 0.803,1.0,-0.942)

$\underline{y}_*$ = (1.046; 1.093,1.809)

Q_* = 5.9760

ΔQ_* = 0.842%

Simulation results are shown in fig.7.3.1.2.

With LDU2 delayed by 5 seconds, the coordinator, LDU1 and LDU2 took 39, 184 and 141 iterations respectively before converging to the solution:

$\underline{s}$ = (0.382; -0.304)

$\underline{V}$ = (0.997; 1.075)

$\underline{c}$ = (-0.616,0.155; 0.830,1.0,-0.942)

$\underline{y}_*$ = (0.995; 1.073,1.865)

Q_* = 5.9869

ΔQ_* = 1.026%

Simulation results are shown in fig.7.3.1.3.

Examining the simulation results shown in fig.7.3.1.1, fig.7.3.1.2 and fig.7.3.1.3 it was found that IPMLF coordination method performed very well even in asynchronous local decision iteration. There was no stability problem associated with this coordination method because IPMLF is a feasible method. Initially, the local decision optimisation problem took about 11 iterations before converging to its solution. However, the rate of convergence was relatively fast when compared with 7 iterations for the synchronous case. The system did not show any sign of instability due to the asynchronous iteration of local decision units. Therefore, basing on the simulation results, conclusion can be made that IPMLF is a very robust coordination method and is extremely useful for on-line control applications.

7.3.2 Interaction Balance Method with Local Feedback

Referring to the simulation results of IBMGF shown in the section 7.2.3, the system coordinated by the Interaction Balance Method was very sensitive to the disturbances imposed on the controls and especially on their interaction inputs. Local decision asynchronisation can be viewed as time delay disturbances on controls and interaction inputs which will jeopardize the stability of the overall system.

During the early study on asynchronous IBMLF, the system was unstable. Tests had been tried by using a first order digital filter with the equation:

$$\underline{u}_*(k) = \alpha \underline{u}_*(k-1) + (1 - \alpha)\ \underline{u}\ (k-1)\ ; \quad \alpha = \text{filter constant}$$

to filter the noise due to LDU' asynchronisation. Using different filter constants ranging from zero to unity, the system failed to converge to its solution.

It was found that by putting a delay (waiting time) in the LDUs before sending the controls to the real subprocesses together with an appropriate local iterative loop gain, the system converged to its solution. Investigation on the values of waiting time, T_w and the gain, $\underline{K}$ used in the local decision optimisation problem with the updating scheme (same as the synchronous IBMLF),

$$\underline{u}_i(k+1) = \underline{u}_i(k) + \underline{K}\ (\underline{u}_{*i}(k) - \underline{u}_i(k))$$

had been conducted. The best waiting time, T_w and gain, $\underline{K}$ was found to be 9 seconds and (0.65; 0.80) respectively. Using these values and with the open-loop solution as the starting point, the coordinator, LDU1 and LDU2 took 54, 300 and 268 iterations respectively before converging to the solution:

$\underline{\lambda}$ = (-2.674, -0.628)
$\underline{c}$ = (-0.741,0.115; 0.923,1.0,-0.974)
$\underline{y}_*$ = (0.947; 1.142,1.940)
Q_* = 5.9621
ΔQ_* = 0.597%

Simulation results are shown in fig.7.3.2.1.

With a different updating scheme used in the local decision optimisation problem,

$$u_{11}(k+1) = u_{11}(k) + K_1(e_1(k) - 2e_2(k))$$
$$u_{21}(k+1) = u_{21}(k) + K_2(e_2(k) - e_1(k))$$

where $e_1(k) = u_{*11}(k) - u_{11}(k)$
$e_2(k) = u_{*21}(k) - u_{21}(k)$

For the gain, $\underline{K}$ = (0.65; 0.80), the search for the new best waiting time had been performed. The best waiting time, T_w was 5 seconds. Using these values and with open-loop solution as the starting point, the coordinator, LDU1 and LDU2 took respectively 35, 438 and 378 iterations before converging to the solution:

$\underline{\lambda}$ = (-2.692, -0.498)
$\underline{c}$ = (-0.719,0.110; 0.890,1.0,-0.996)
$\underline{y}_*$ = (0.991; 1.147,1.907)
Q_* = 5.9629
ΔQ_* = 0.623%

Simulation results are shown in fig.7.3.2.2.

Basing on the simulation results obtained, it was found that the IBMLF was very sensitive to the asynchronous local iteration of the decision units. This is because Interaction Balance Method is a non-feasible method and the determination of the optimal performance index for each decision unit relies heavily on the controls and the interaction variables of other decision

units. Without synchronisation between decision units simply implied that the connections between each decision unit were being cut off which would cause instabilty. By choosing the gain $\underline{K}$ = (0.65; 0.80) in the local decision updating equation together with an appropriate waiting time, instability due to asynchronous iteration of local decision units could be stabilised and the system converged to its solution. It was found that there was a minimum waiting time that the system could be stabilised for a given set of gain values.

7.4 Summary of the Simulation Results

Using the open-loop solution set as the first estimate for the coordination parameters, the following simulation results had been obtained for synchronous local iteration which were summaried as follows:

Coordination method	Performance index	Suboptimality (%)	Iterations required	Iteratio time (s)
IBMGF	5.989	0.986	6	196
IBMLF	5.962	0.613	33	4910
IPMGF	5.933	0.110	32	950
IPMLF	5.975	0.821	43	2386

For the asynchronous IBMLF and IPMLF, similar results were obtained as the synchronous cases. However, the rate of convergence of the local decision problems were slower, especially for the IBMLF. When stability of the local decision problem was concerned, the IPMLF performed far more better than the IBMLF. Stabilisation was required for the asynchronous IBMLF.

7.5 Graphical Output of the Simulation Results

The following figures are the graphical output of the simulation results for synchronous and asynchronous local decision iteration.

Synchronous Local Decision Iteration

Fig.7.2.1.1. IPMGF simulation result: with open-loop solution as the starting point.

Fig.7.2.1.2. IPMGF simulation result: with zero as the starting point.

Fig.7.2.1.3. IPMGF simulation result: modelling disturbances in controls.

Fig.7.2.1.4. IPMGF simulation result: modelling LDUs failure.

Fig.7.2.1.5. IPMGF simulation result: modelling interconnections failure.

Fig.7.2.2.1. IPMLF simulation result: with open-loop solution and zero shift vector as the starting point.

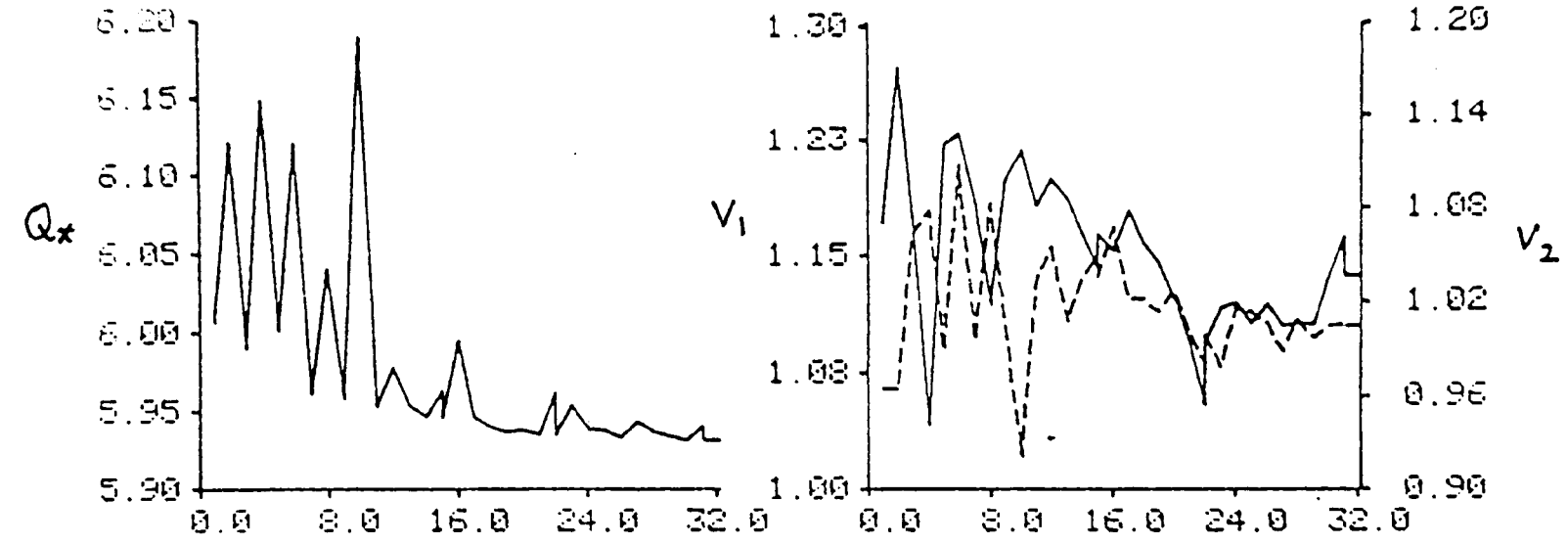
Fig.7.2.3.1. IBMGF simulation result: with zero as the starting point.

Fig.7.2.3.2. IBMGF simulation result: modelling disturbances in controls.

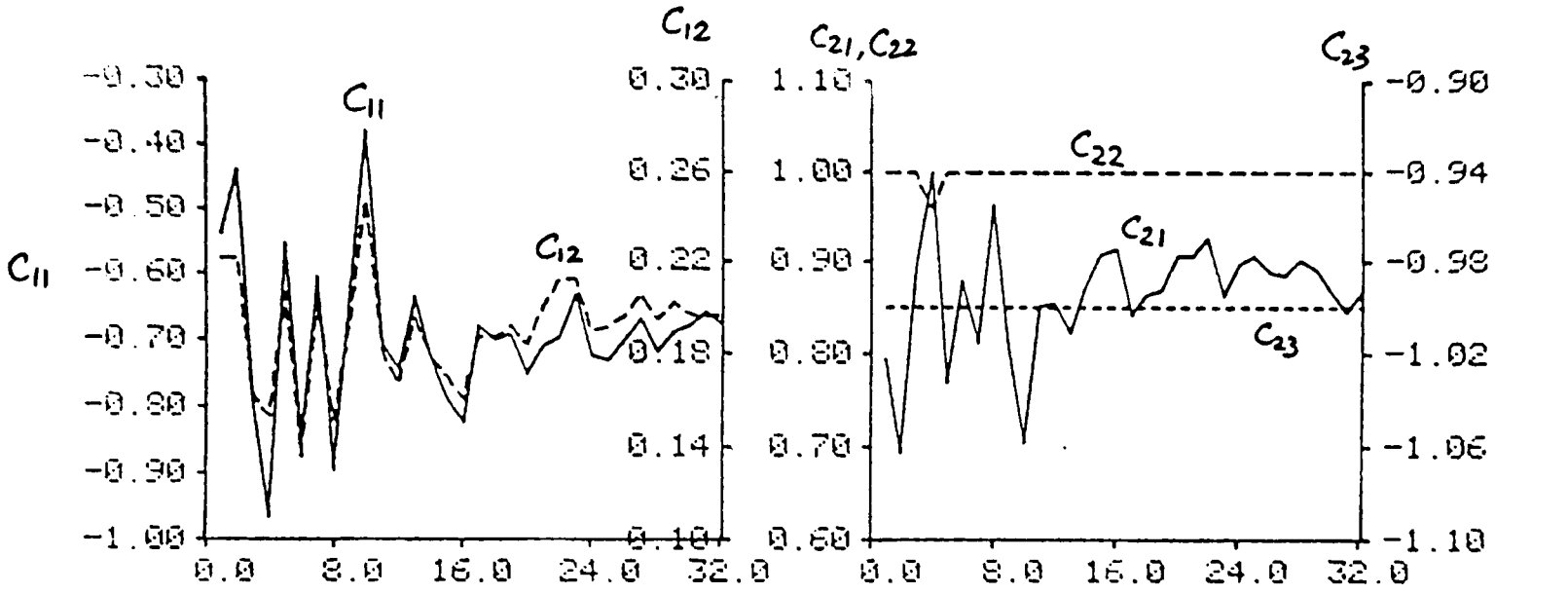
Fig.7.2.3.3. IBMGF simulation result: modelling disturbances in LDUs.

Fig.7.2.3.4. IBMGF simulation result: modelling interconnection failures - test 1.

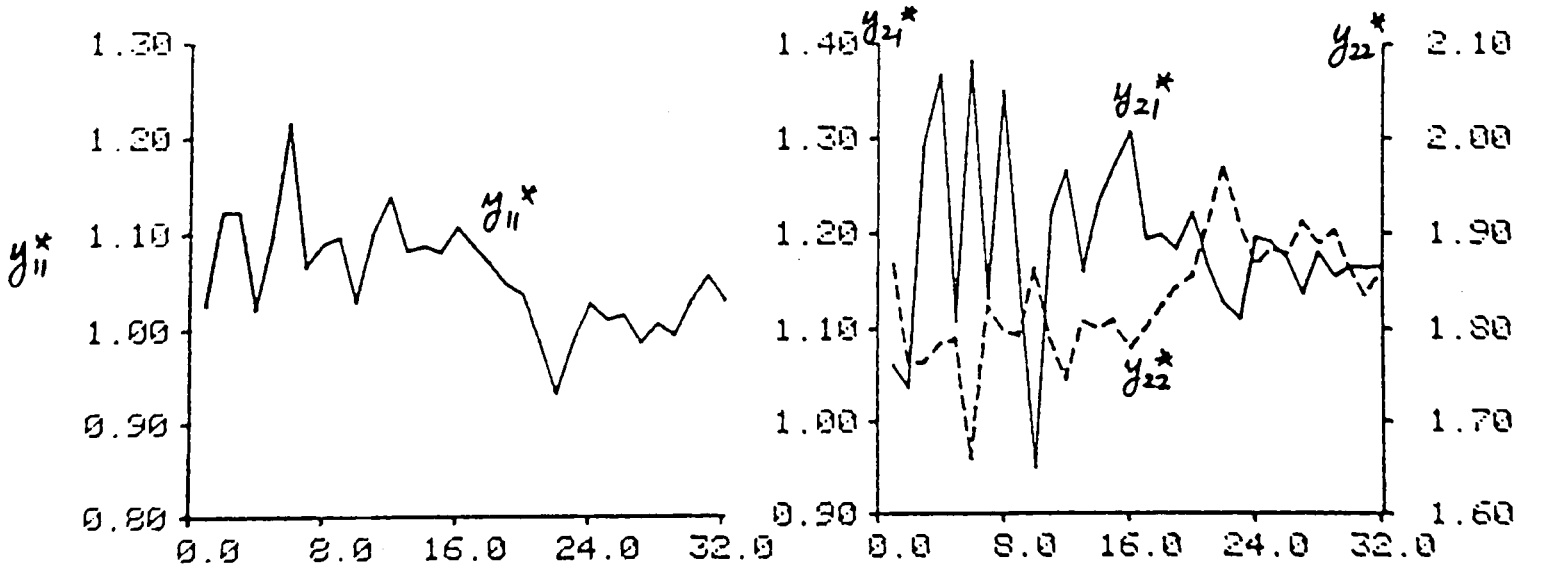
Asychronous Local Decision Iteration



a) Real performance index b) Interaction variables

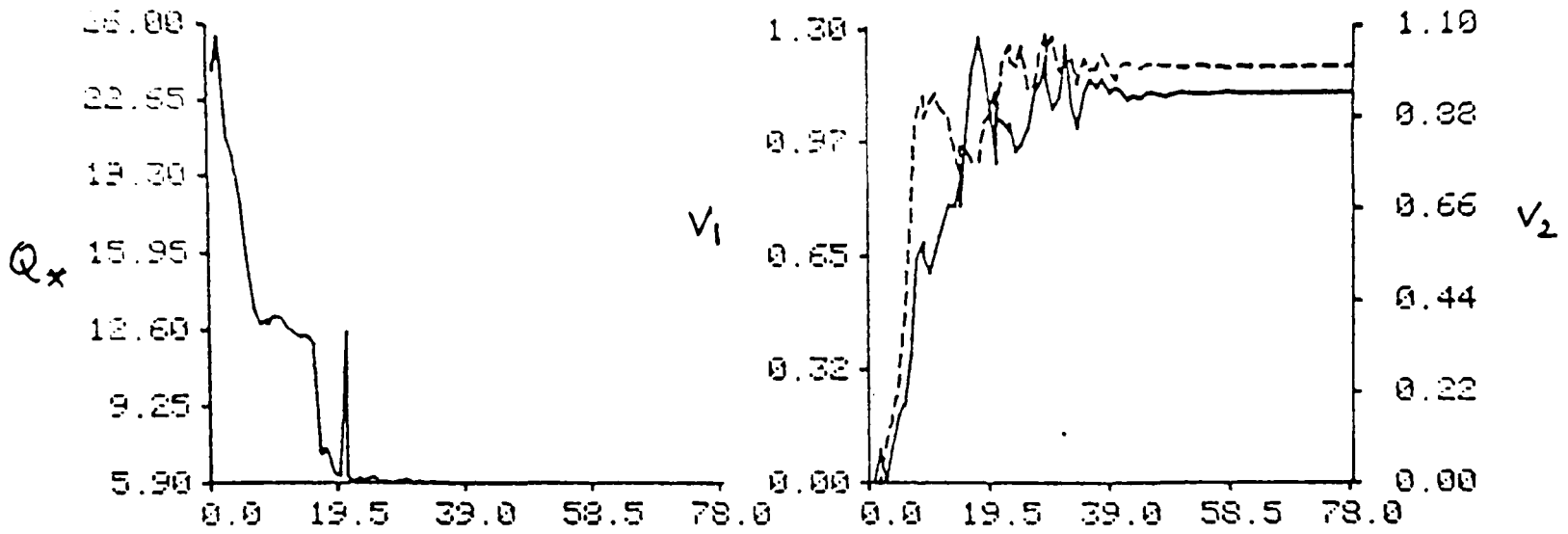


c) LDU1 controls d) LDU2 controls

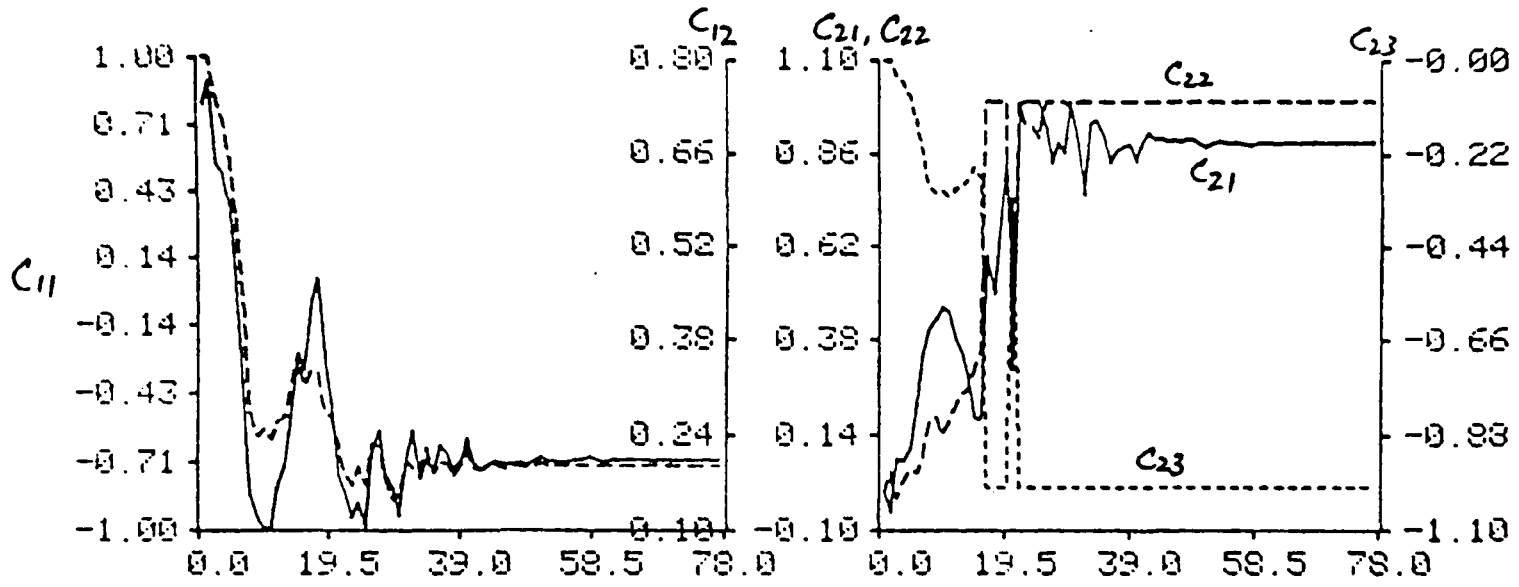


e) LDU1 real output f) LDU2 real outputs

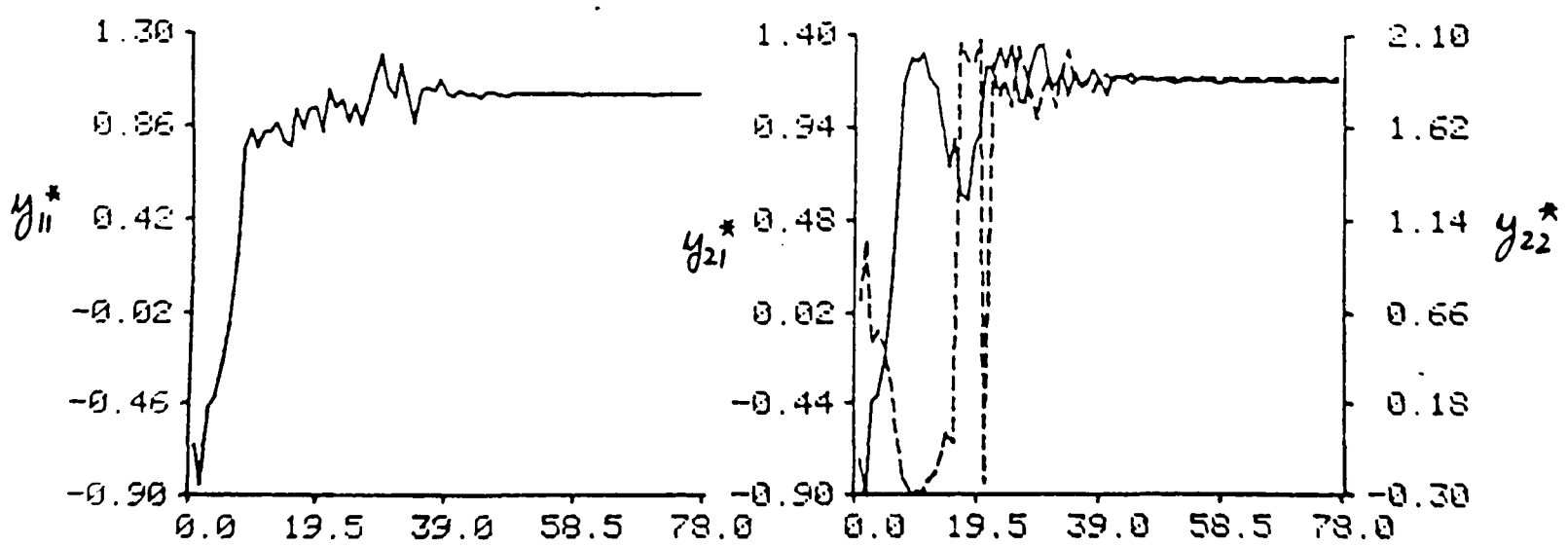
Fig.7.2.1.1. IPMGF simulation result: with open-loop solution as the starting point.



a) Real performance index b) Interaction variables

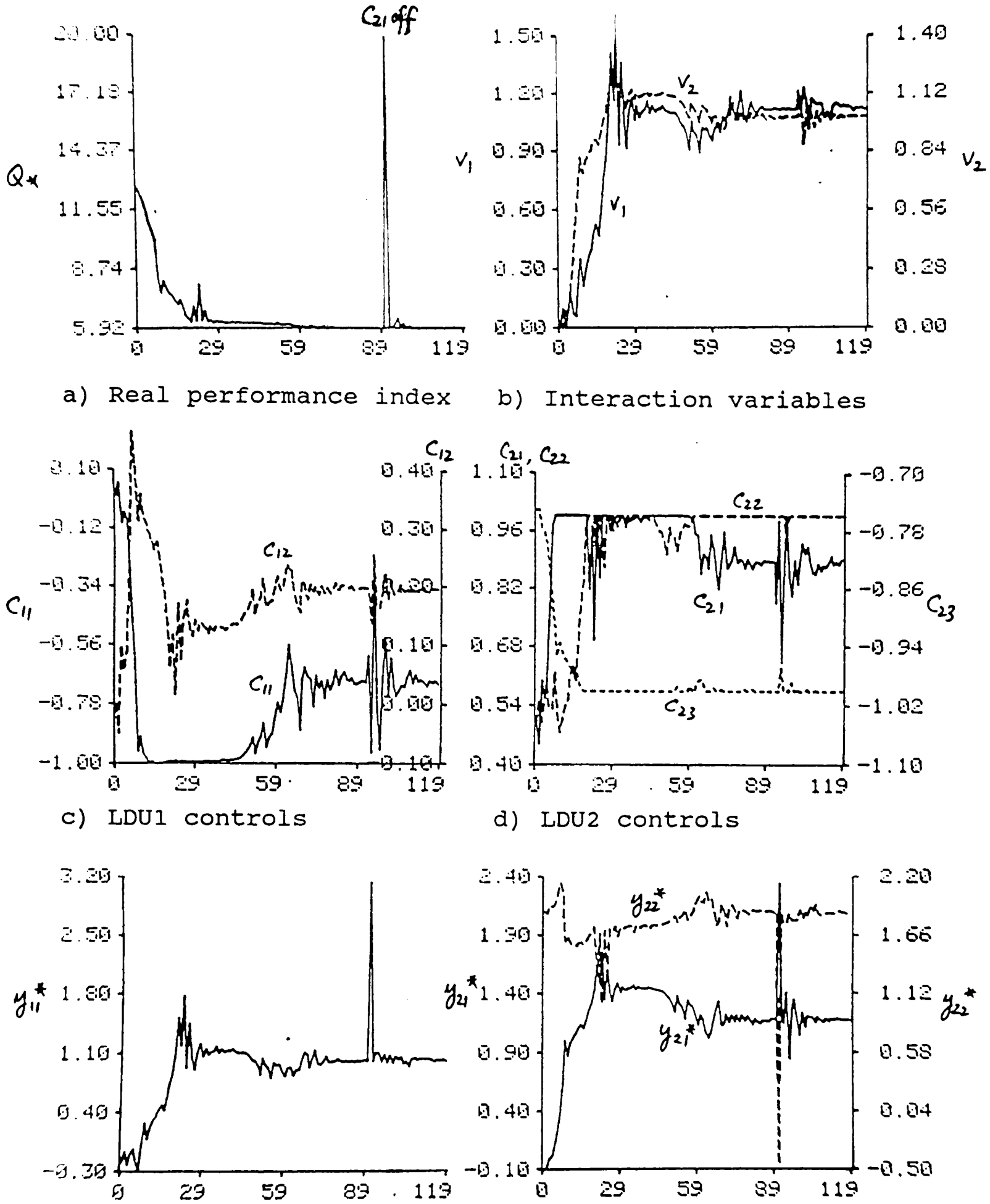


c) LDU1 controls d) LDU2 controls



e) LDU1 real output f) LDU2 real outputs

Fig.7.2.1.2. IPMGF simulation result: with zero as the starting point.



a) Real performance index

b) Interaction variables

c) LDU1 controls

d) LDU2 controls

e) LDU1 real output

f) LDU2 real outputs

Fig.7.2.1.3. IPMGF simulation result: modelling disturbances in controls

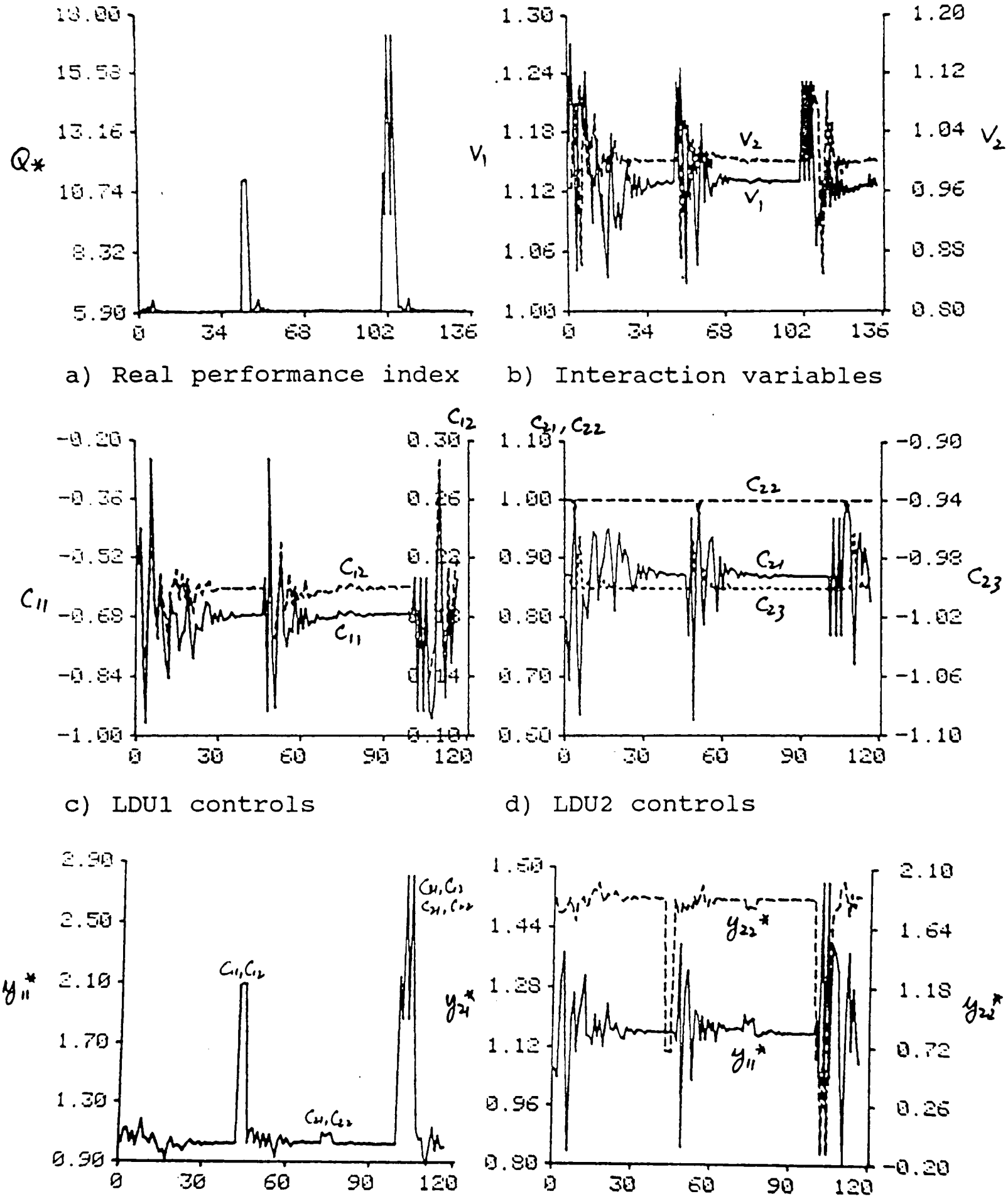
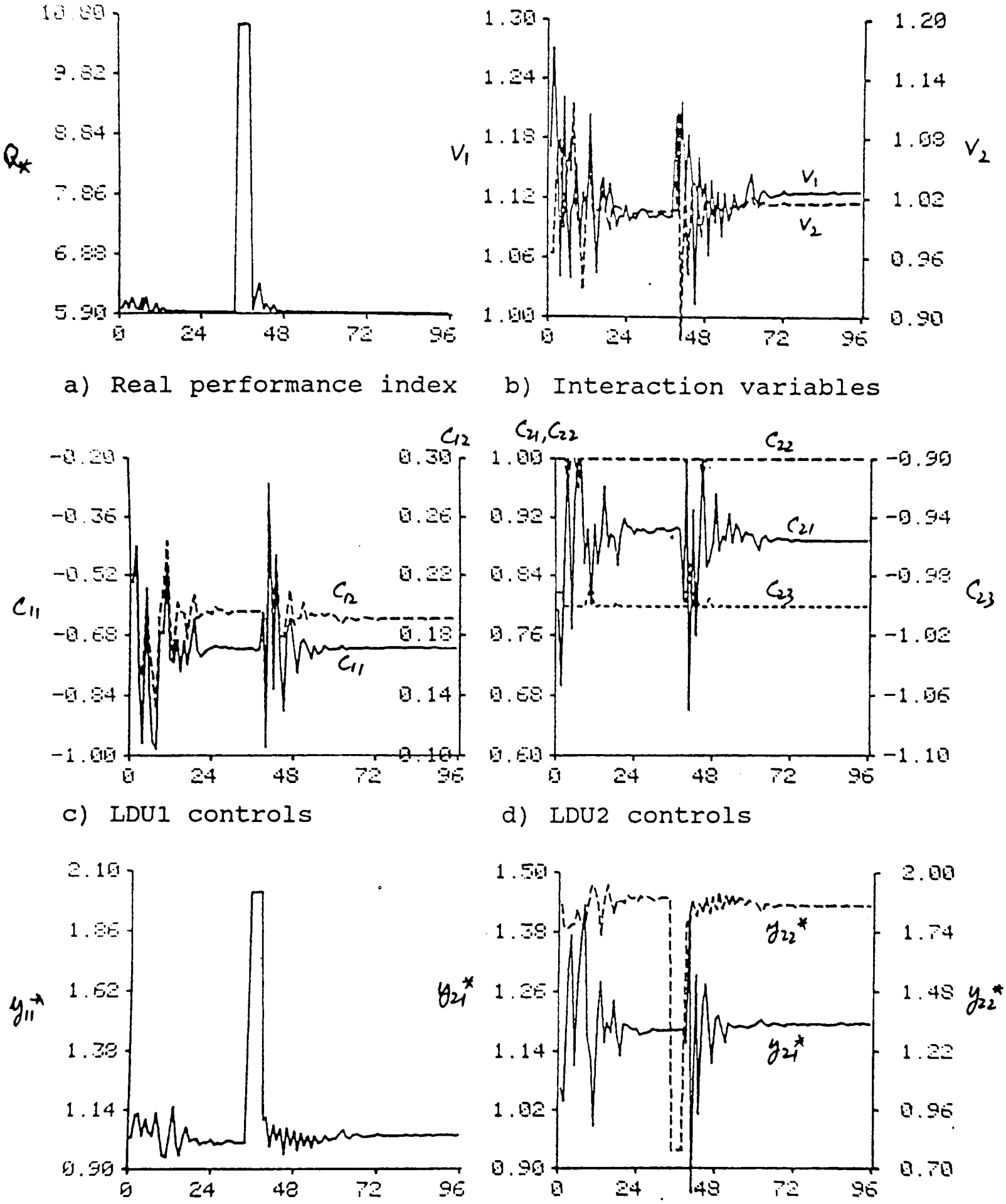


Fig.7.2.1.4. IPMGF simulation result: modelling LDUs failure



a) Real performance index

b) Interaction variables

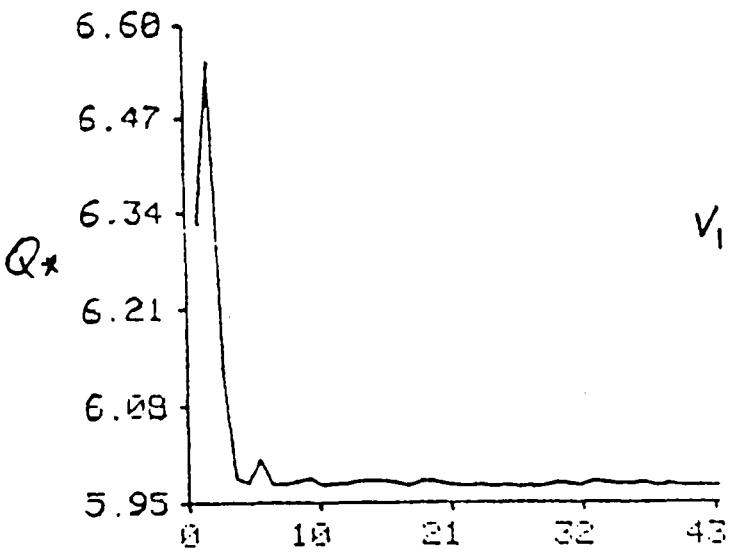
c) LDU1 controls

d) LDU2 controls

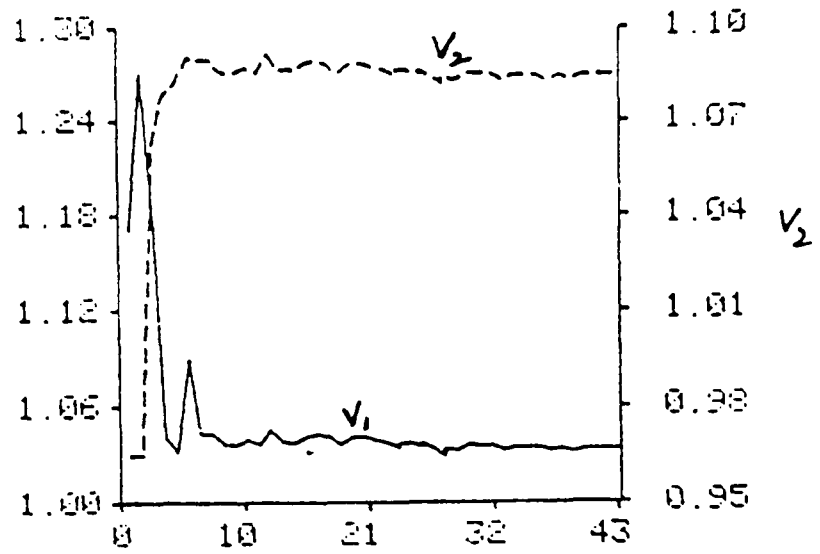
e) LDU1 real output

f) LDU2 real outputs

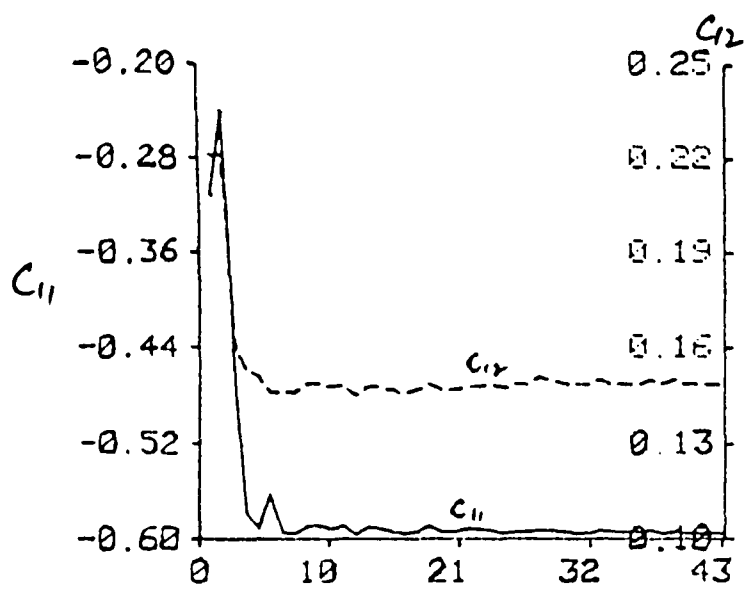
Fig.7.2.1.5. IPMGF simulation result: modelling interconnections failure



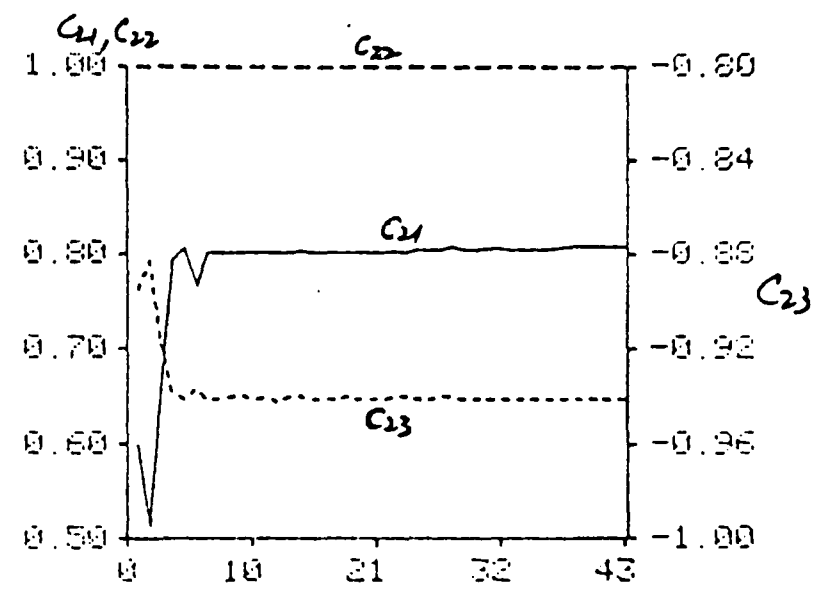
a) Real performance index



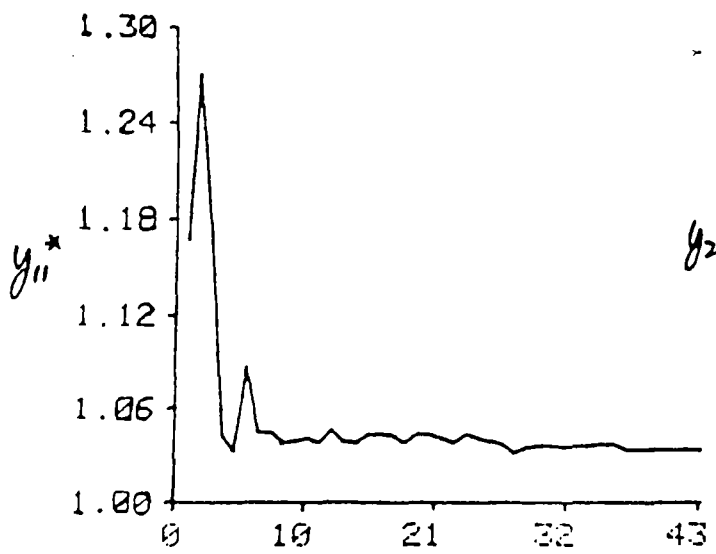
b) Interaction variables



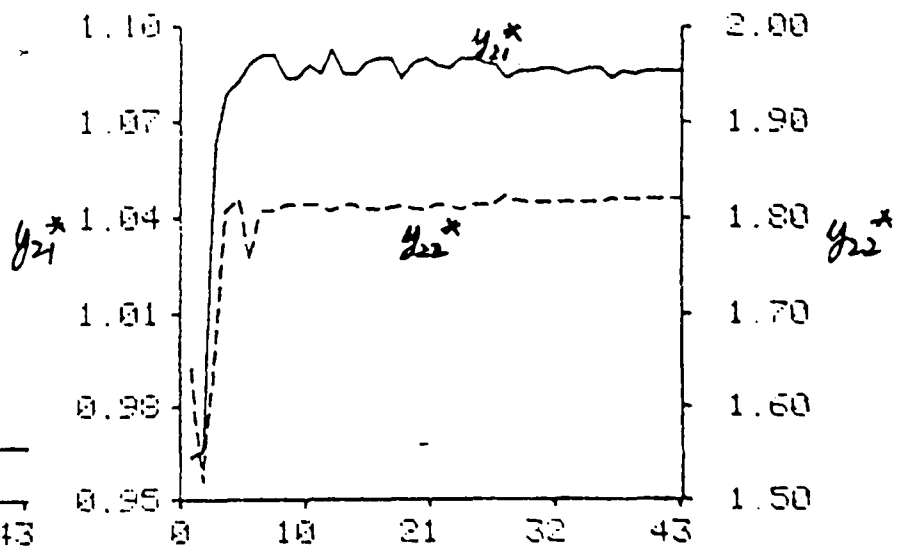
c) LDU1 controls



d) LDU2 controls

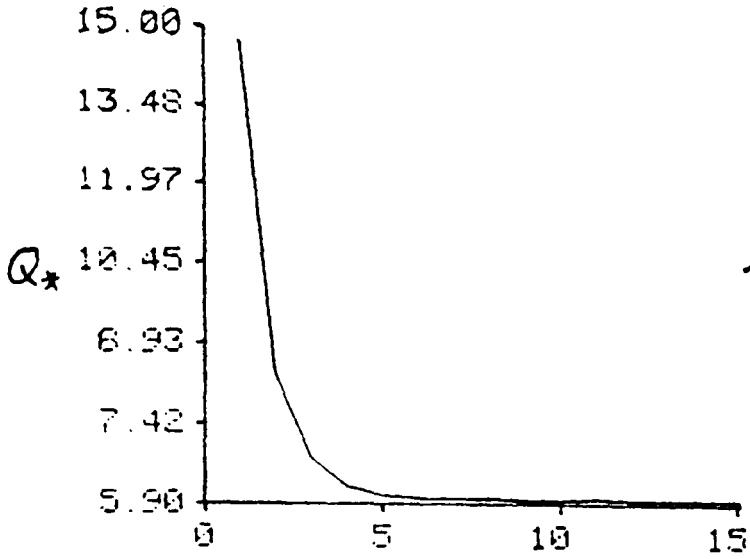


e) LDU1 real output

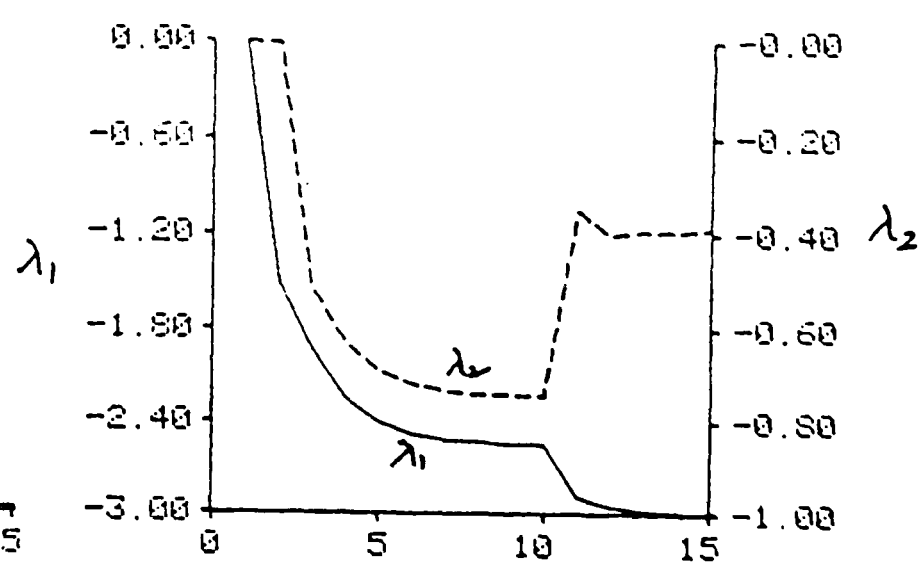


f) LDU2 real outputs

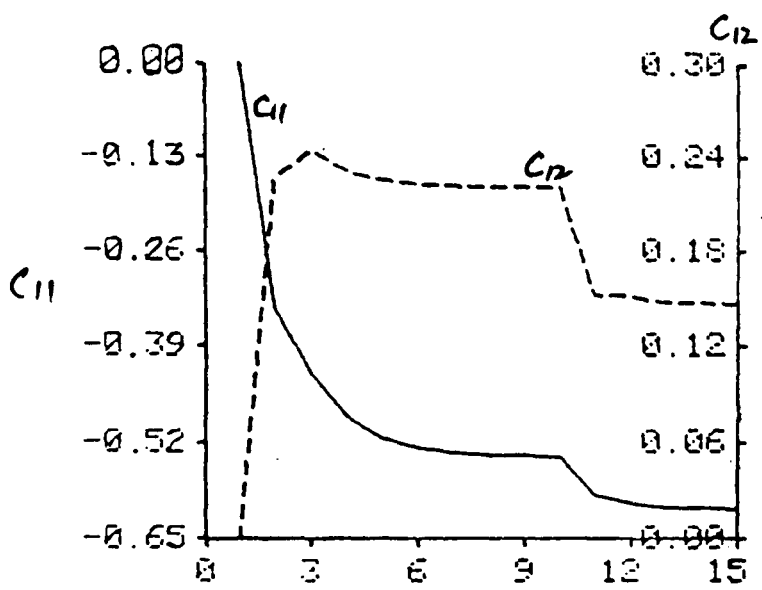
Fig.7.2.2.1. IPMLF simulation result: with open-loop solution and zero shift vector as the starting point.



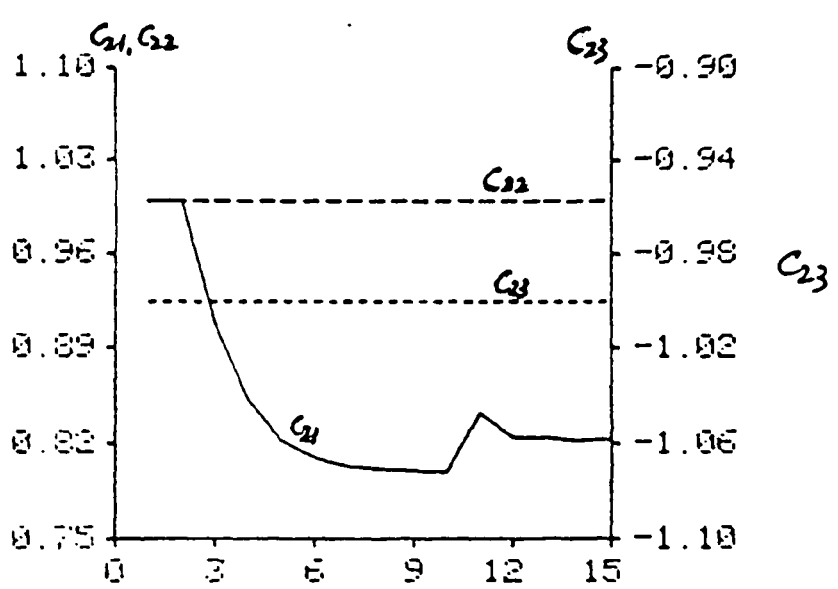
a) Real performance index



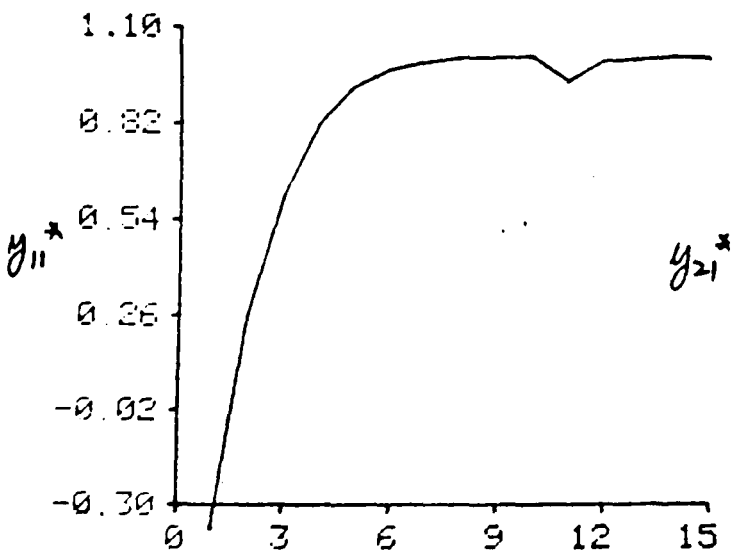
b) Lagrange multipliers



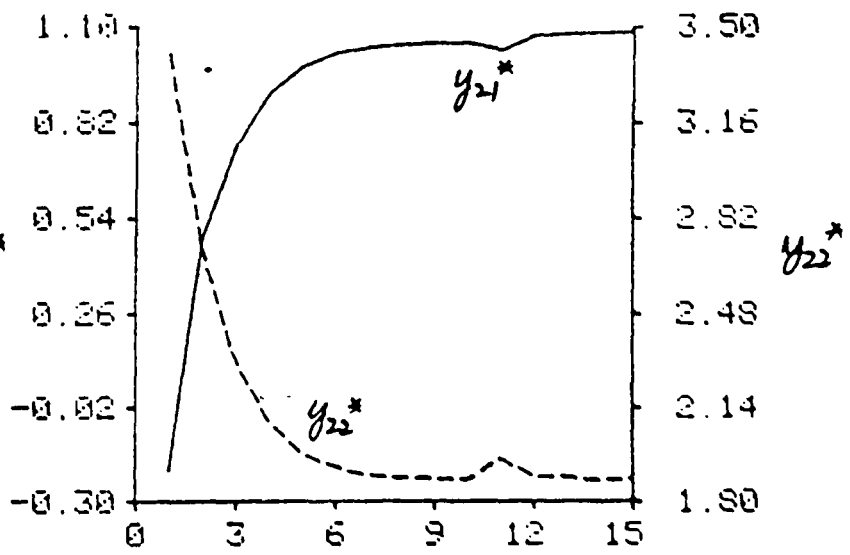
c) LDU1 controls



d) LDU2 controls

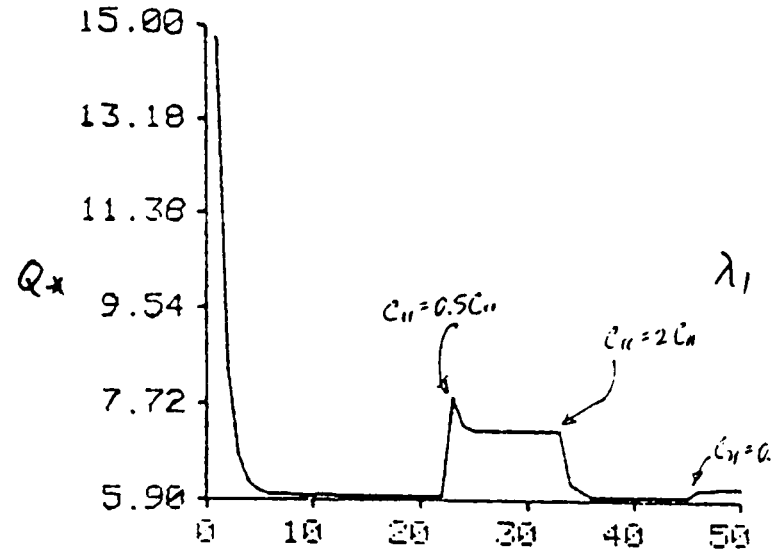


e) LDU1 real output

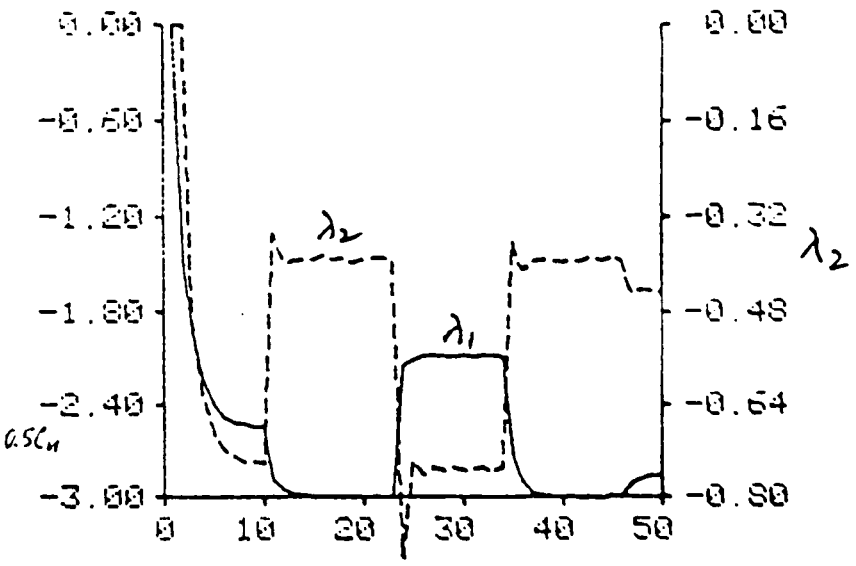


f) LDU2 real outputs

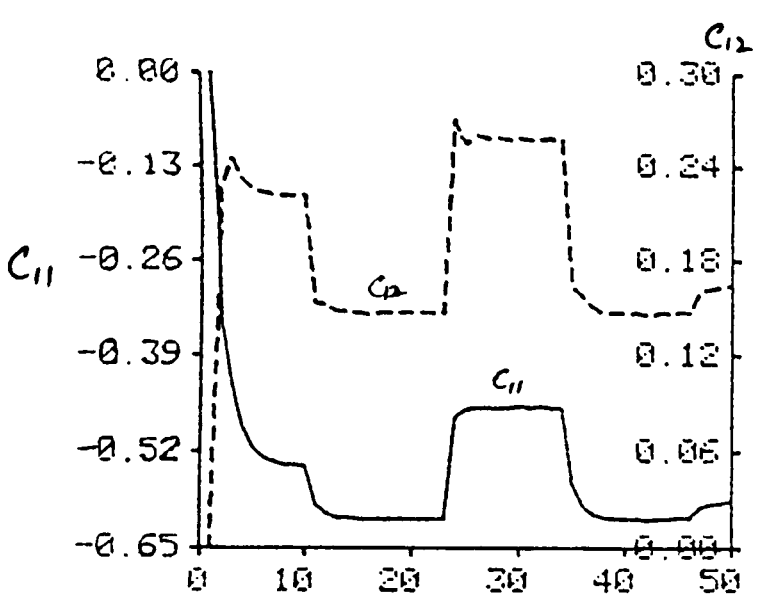
Fig.7.2.3.1. IBMGF simulation result: with zero as the starting point.



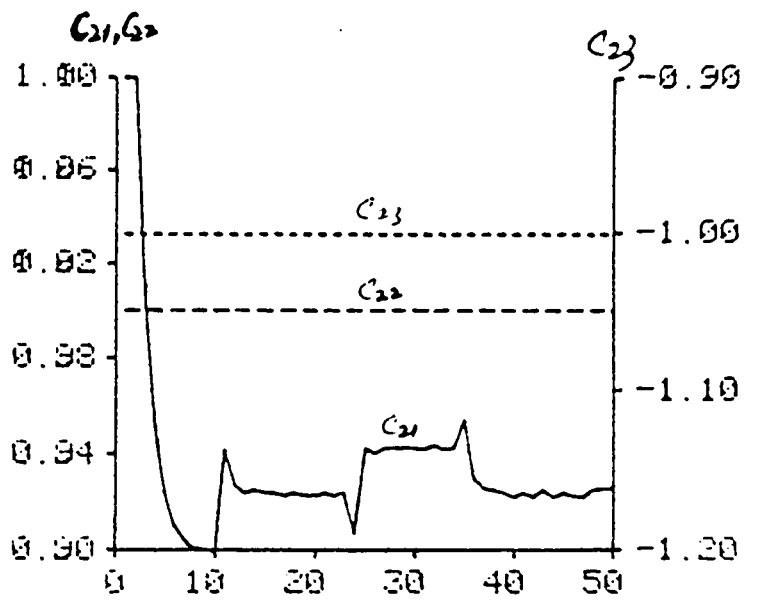
a) Real performance index



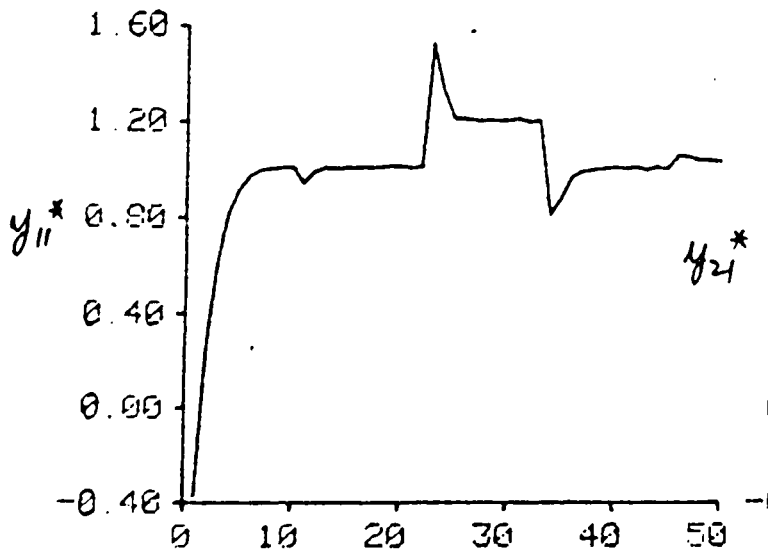
b) Lagrange multipliers



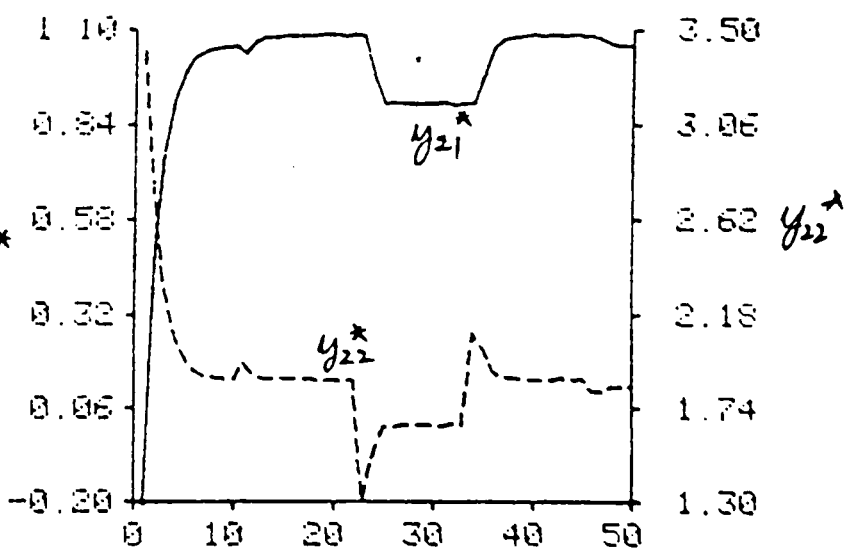
c) LDU1 controls



d) LDU2 controls



e) LDU1 real output



f) LDU2 real outputs

Fig.7.2.3.2. IBMGF simulation result: modelling disturbances in controls.

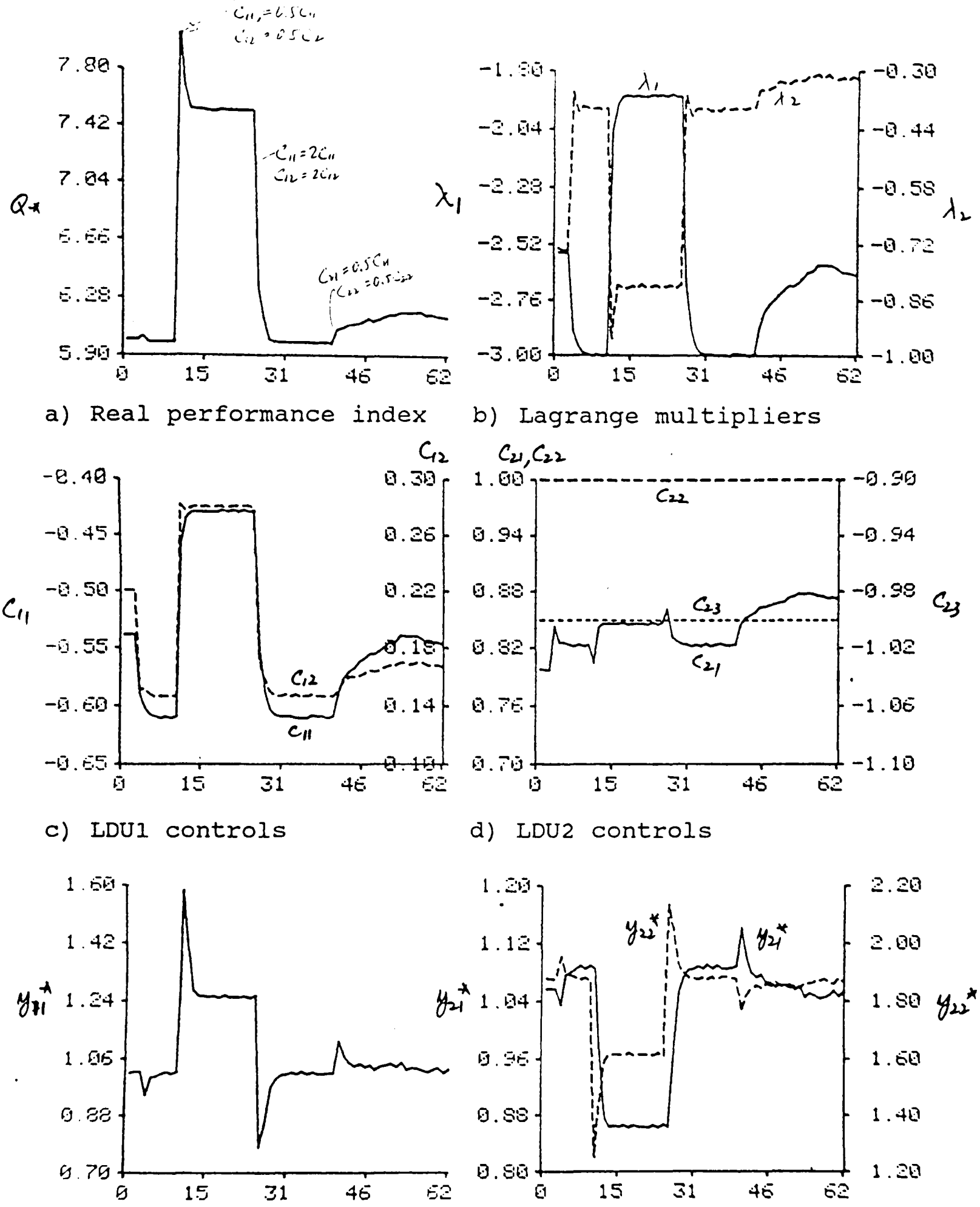
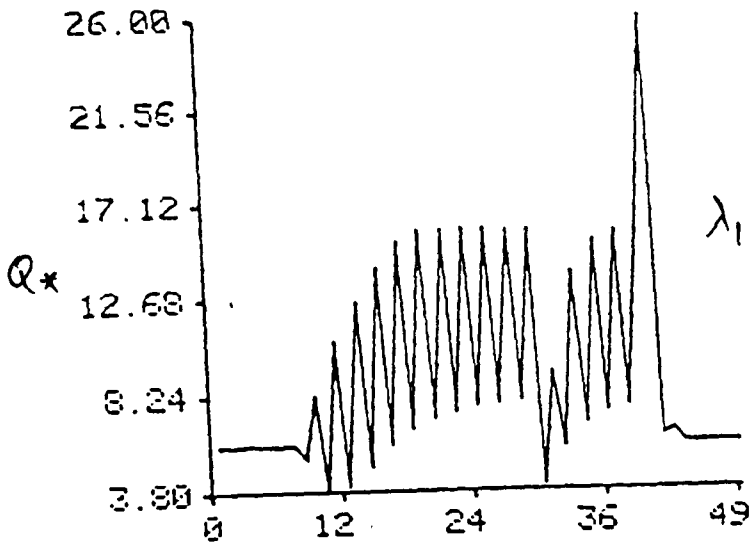
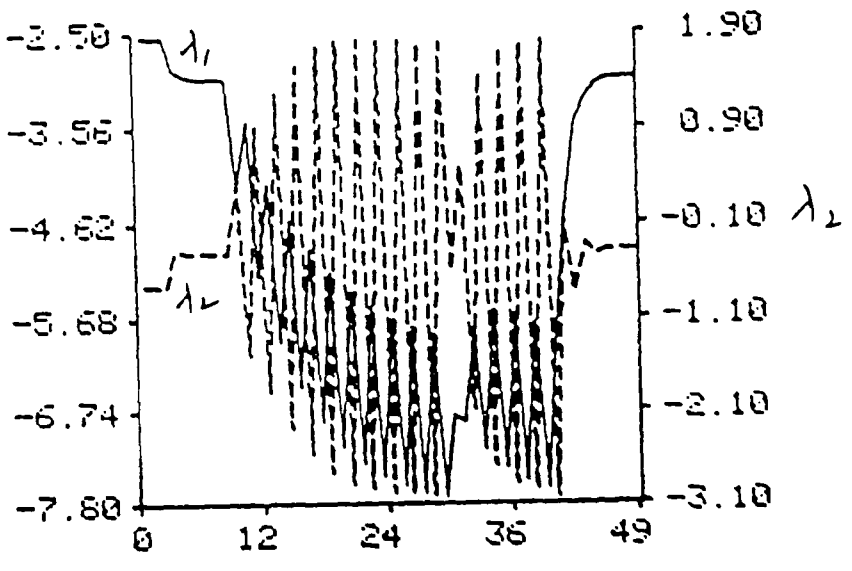


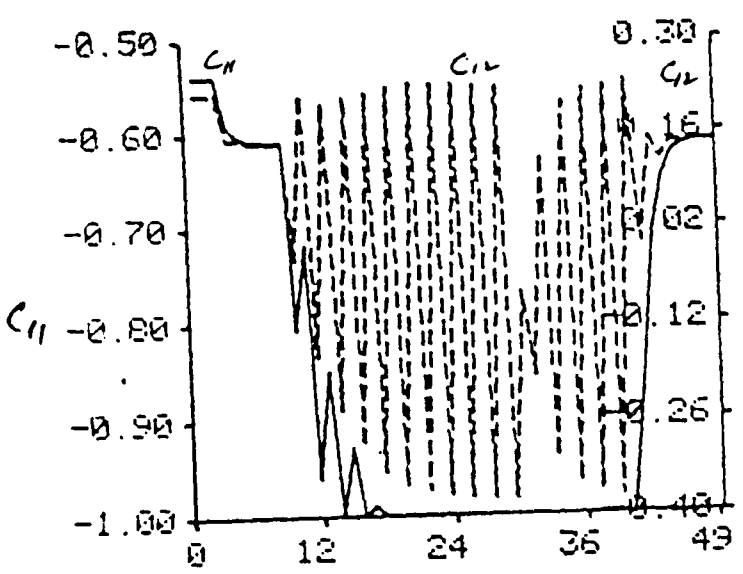
Fig.7.2.3.3. IBMGF simulation result: modelling disturbances in LDUs.



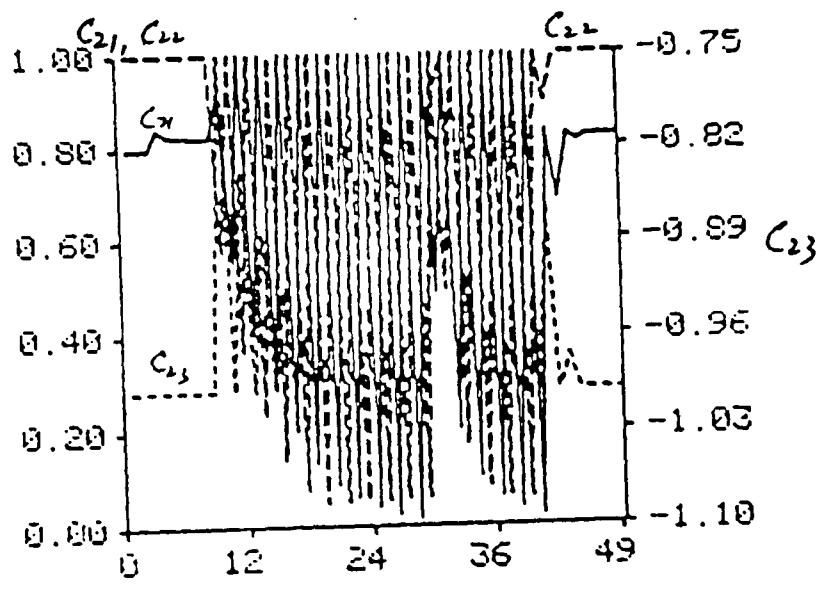
a) Real performance index



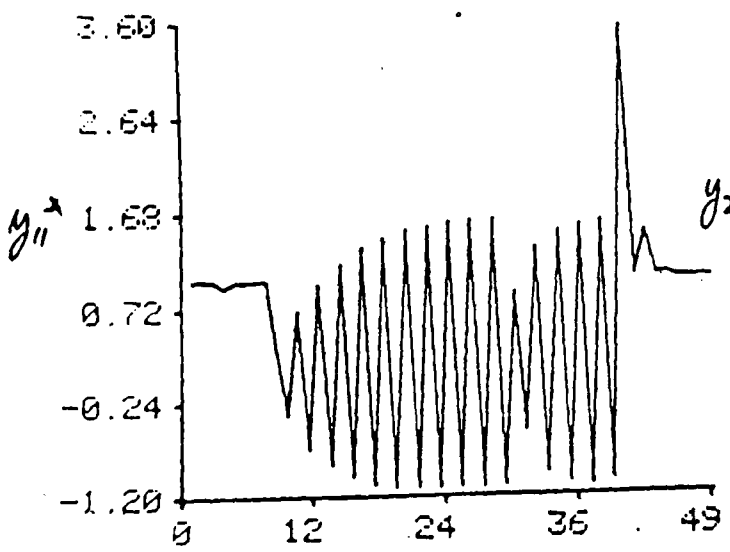
b) Lagrange multipliers



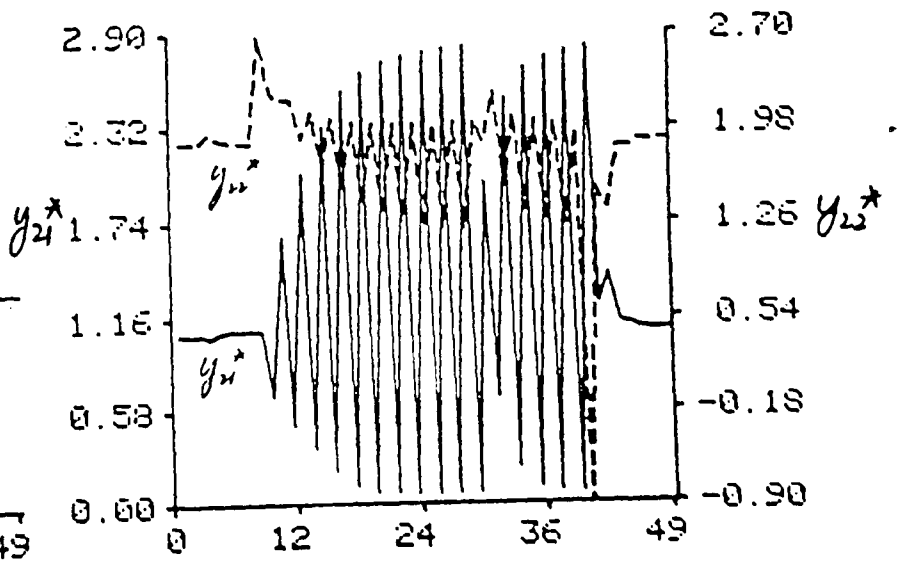
c) LDU1 controls



d) LDU2 controls

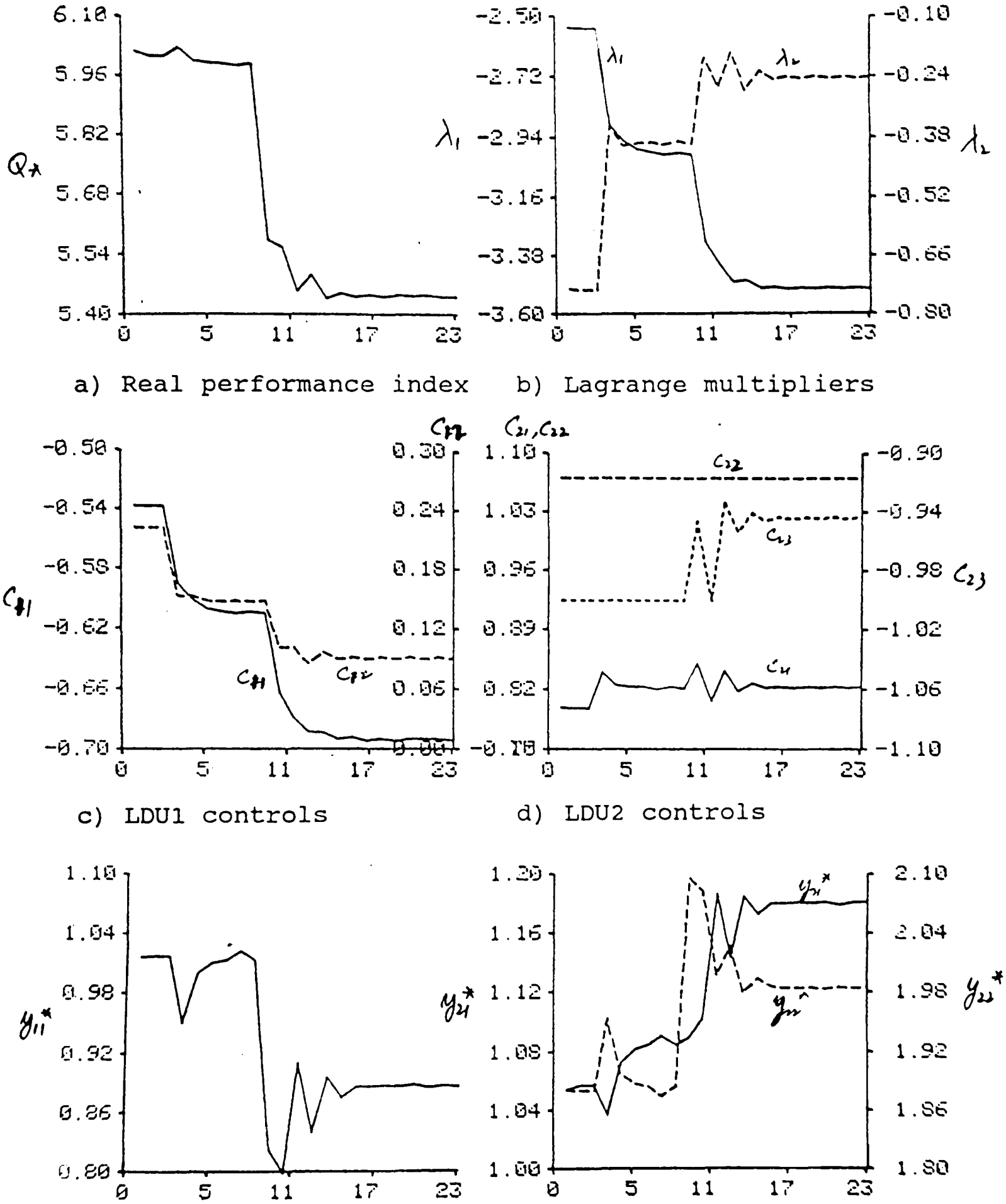


e) LDU1 real output



f) LDU2 real outputs

Fig.7.2.3.4. IBMGF simulation result: modelling interconnection failures - test 1.



a) Real performance index

b) Lagrange multipliers

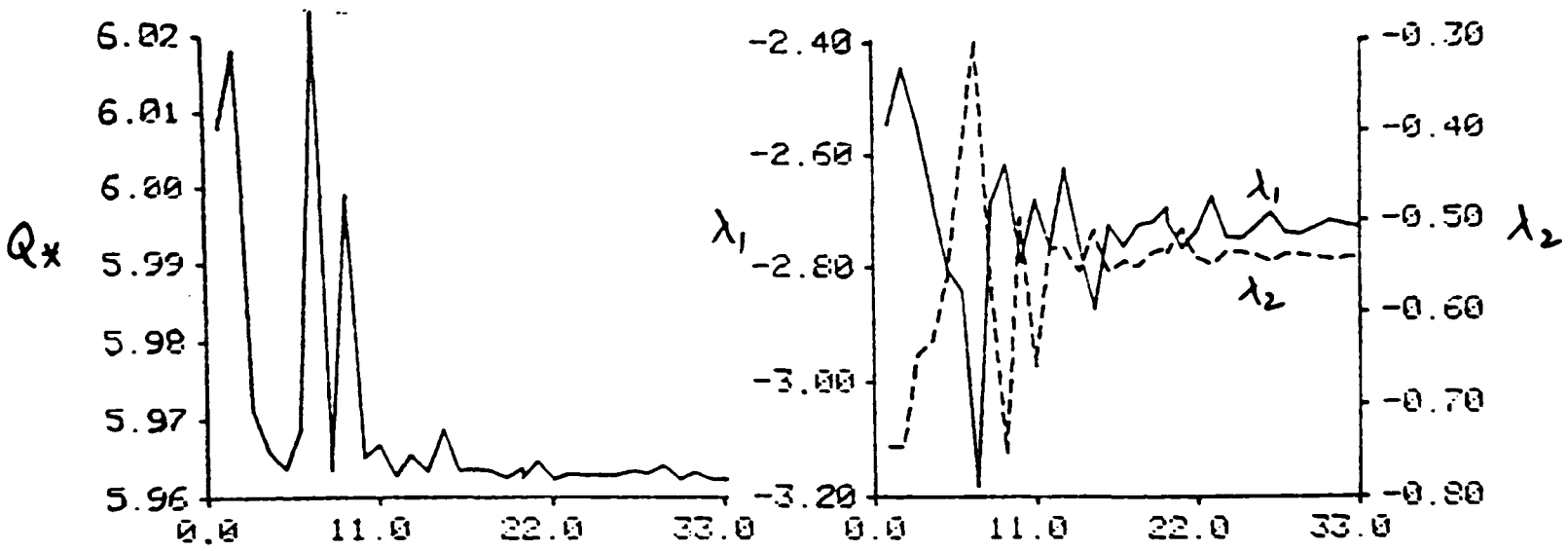
c) LDU1 controls

d) LDU2 controls

e) LDU1 real output

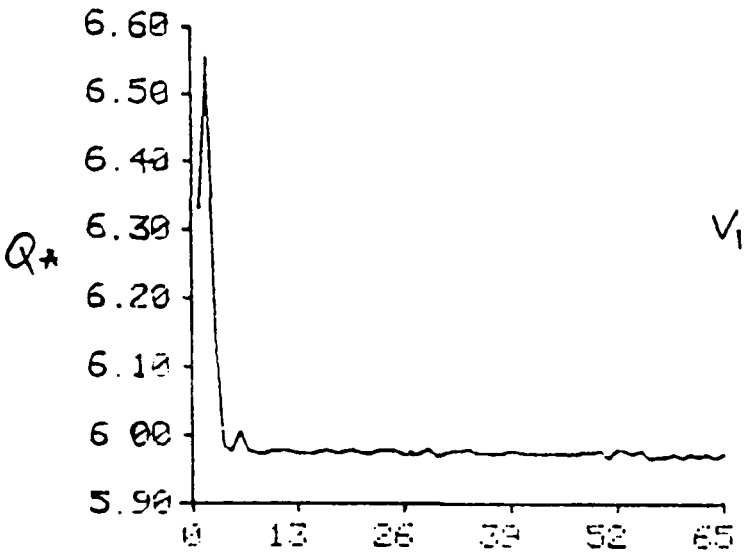
f) LDU2 real outputs

Fig.7.2.3.5. IBMGF simulation result: modelling interconnection failures - test 2.

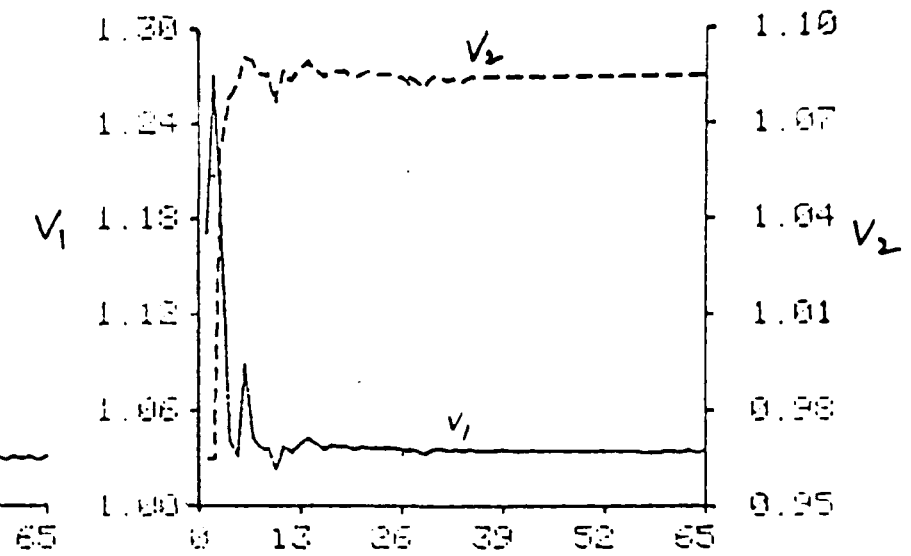


a) Real performance index b) Lagrange multipliers

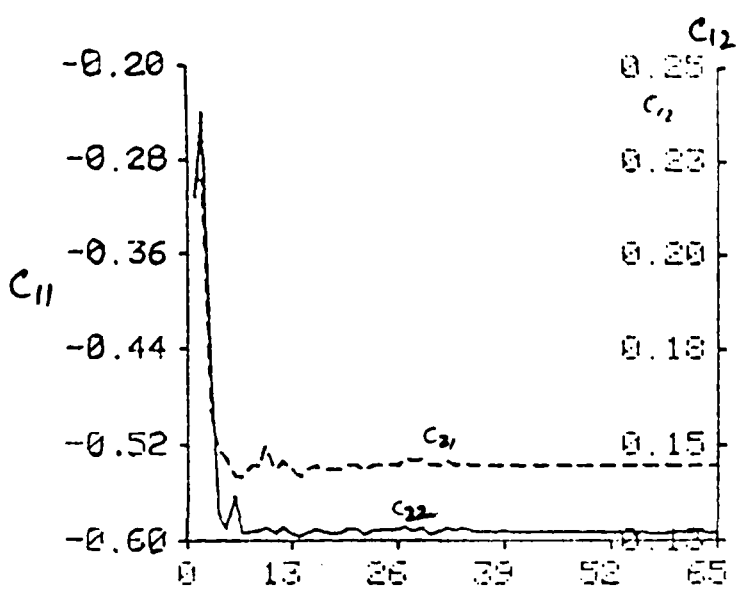
Fig.7.2.4.1. IBMLF simulation result: with open-loop solution as the starting point.



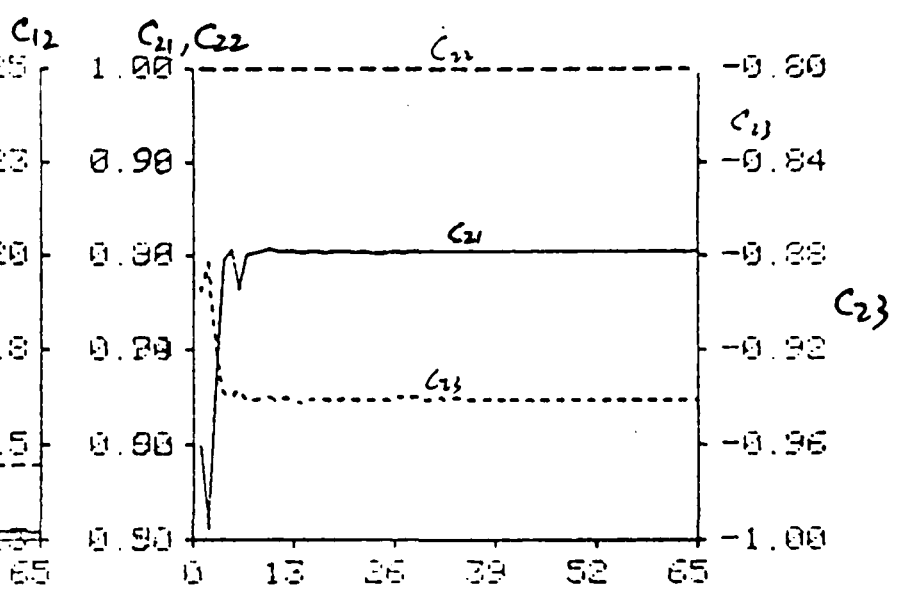
a) Real performance index



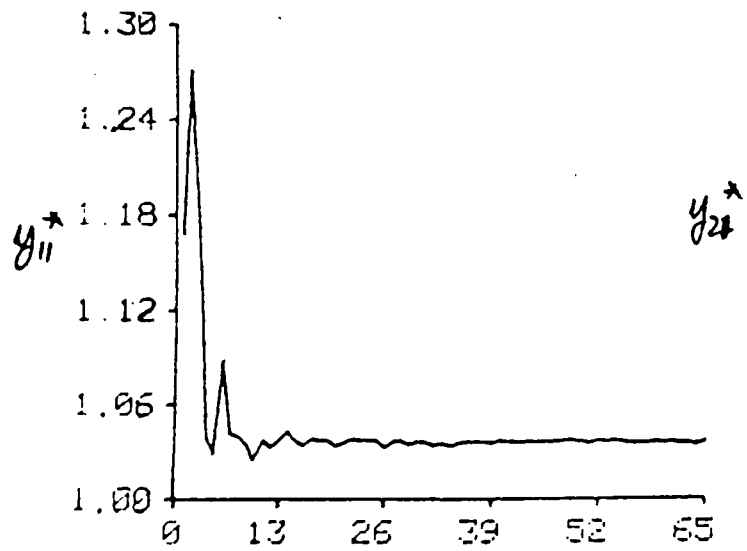
b) Interaction variables



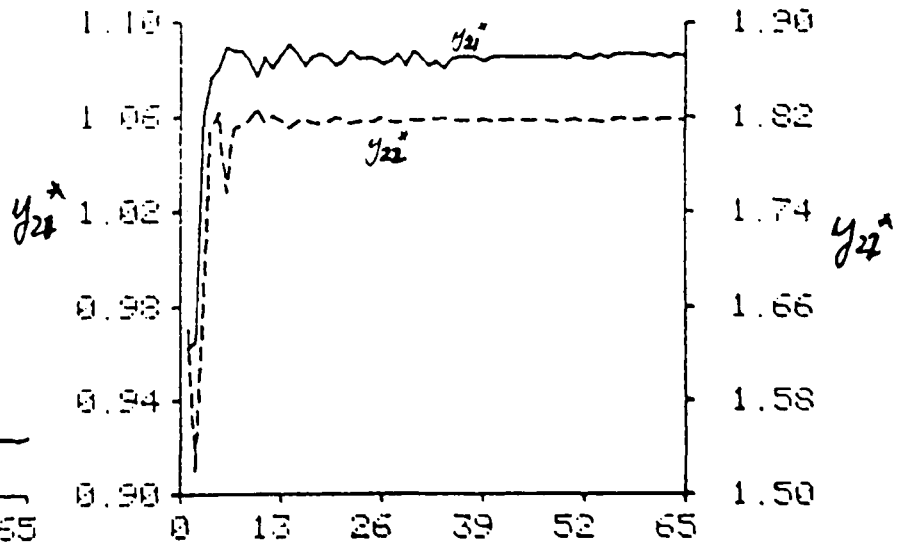
c) LDU1 controls



d) LDU2 controls

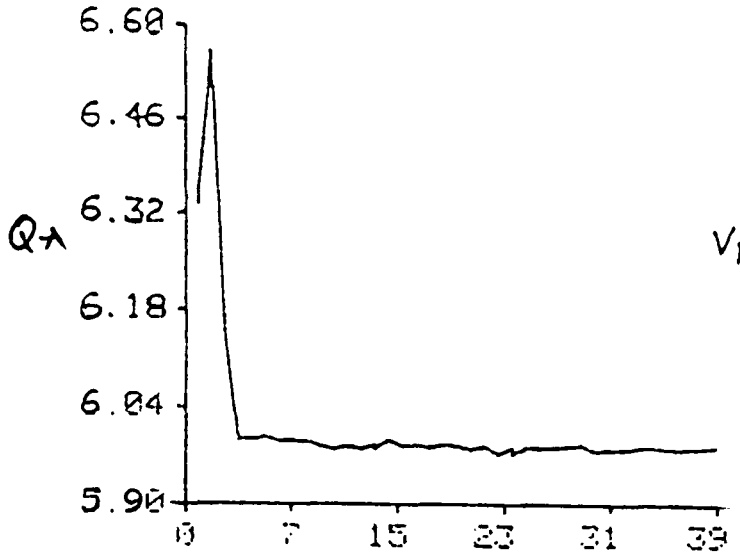


e) LDU1 real output

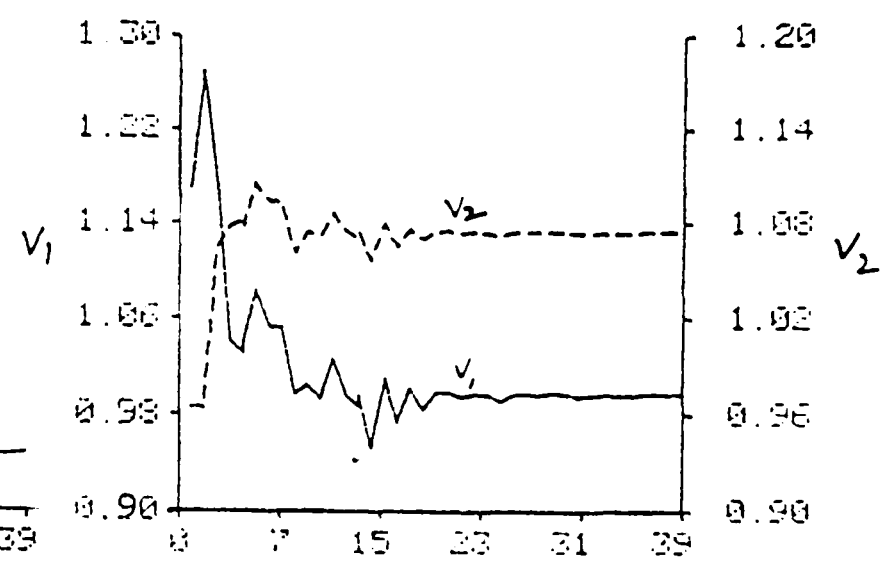


f) LDU2 real outputs

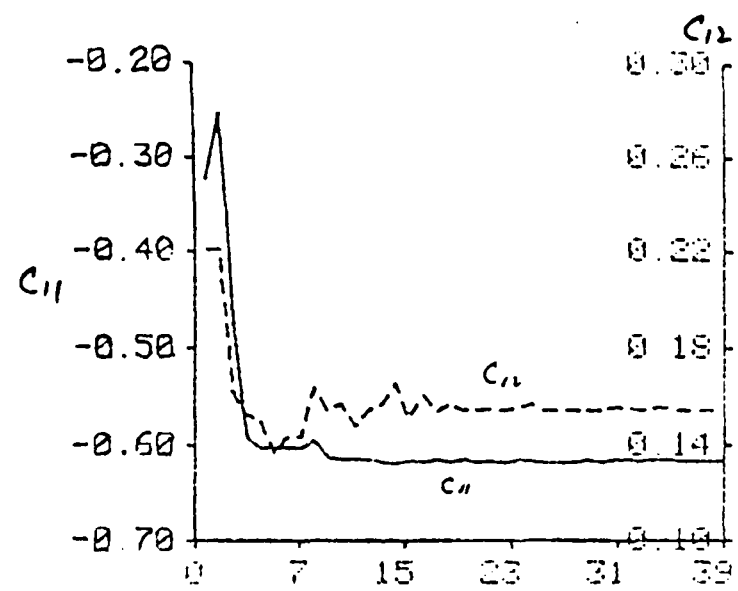
Fig.7.3.1.1. Asynchronous IPMLF simulation result: with open-loop solution and zero shift vector as the starting point.



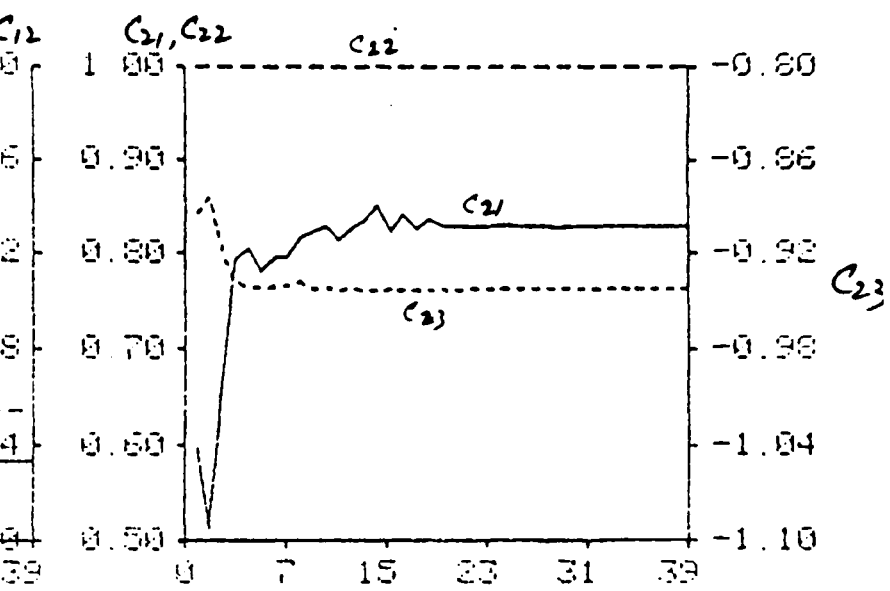
a) Real performance index



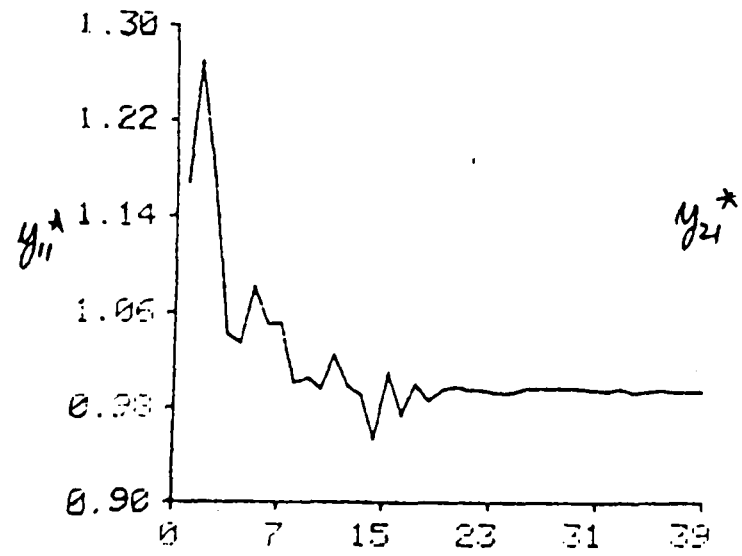
b) Interaction variables



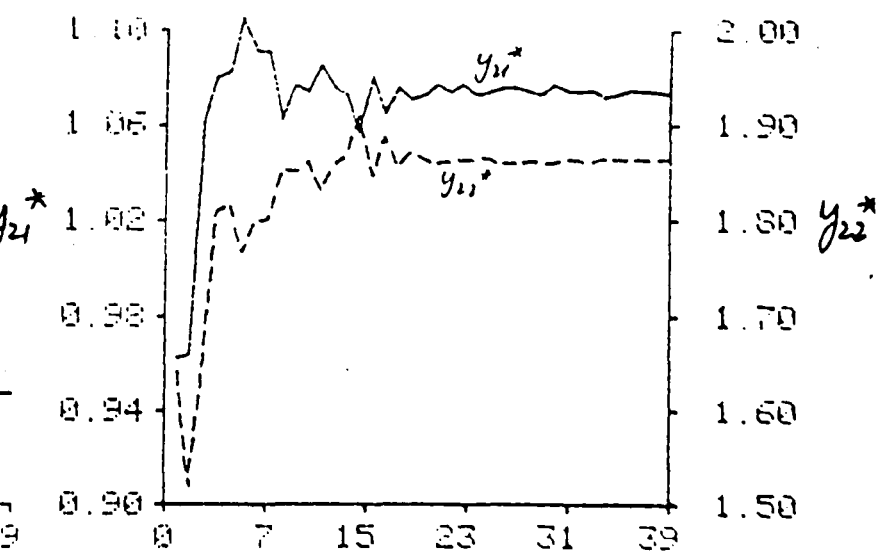
c) LDU1 controls



d) LDU2 controls

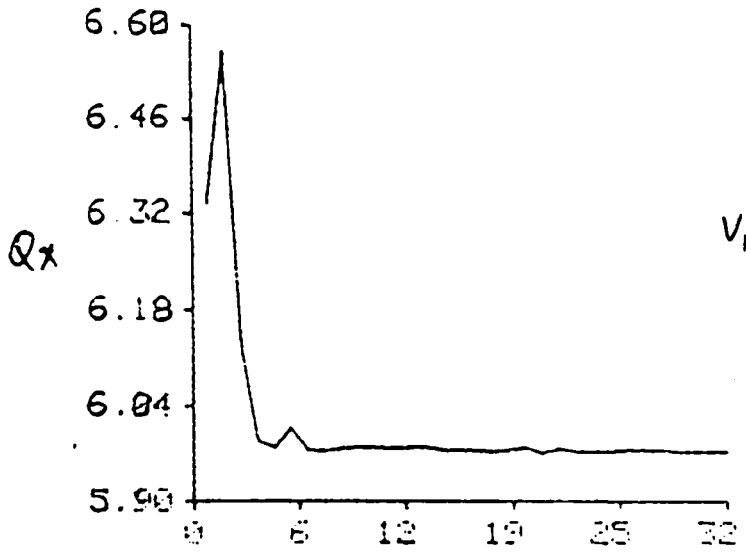


e) LDU1 real output

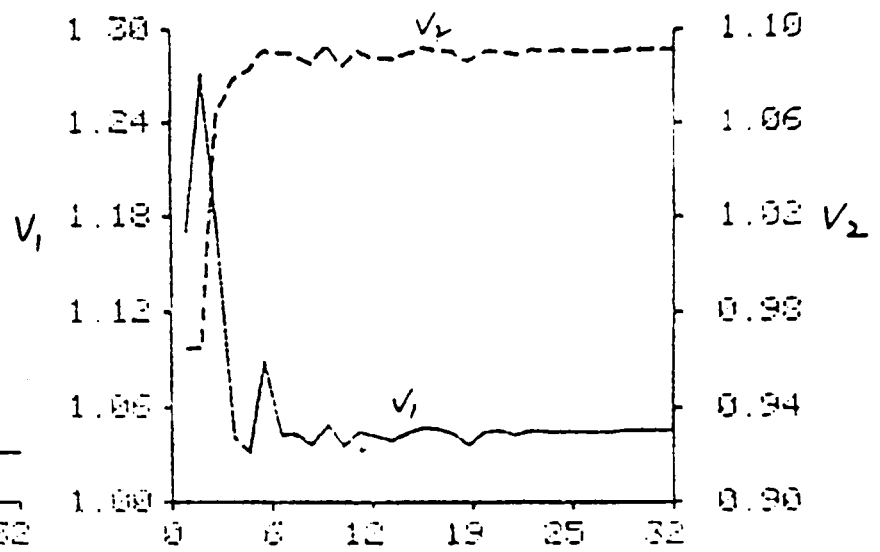


f) LDU2 real outputs

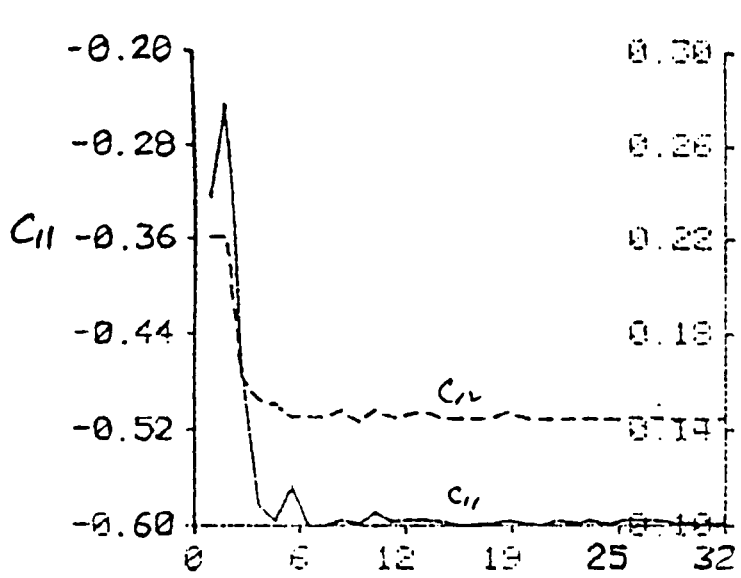
Fig.7.3.1.2. Asynchronous IPMLF simulation result: with open-loop solution and zero shift vector as the starting point and LDU1 delayed by 5 seconds.



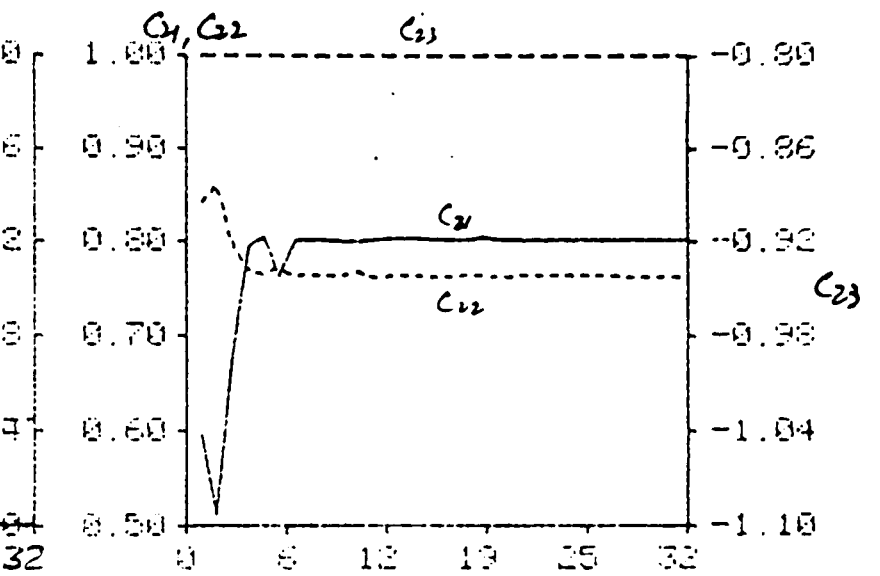
a) Real performance index



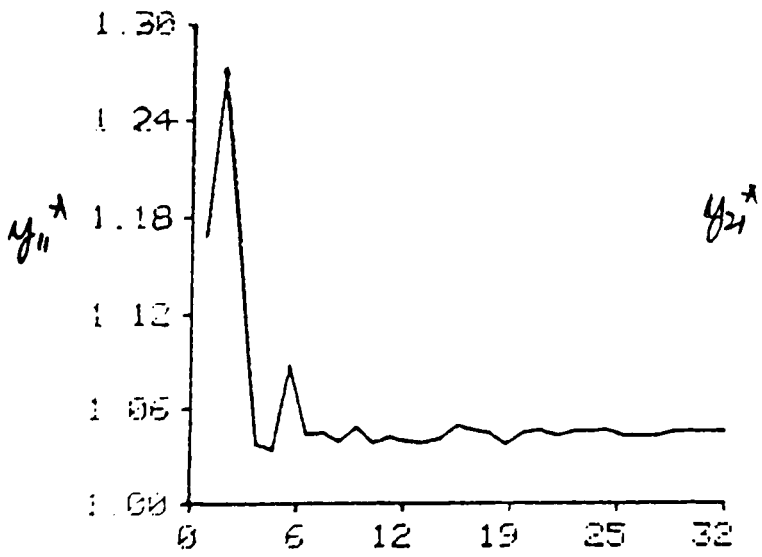
b) Interaction variables



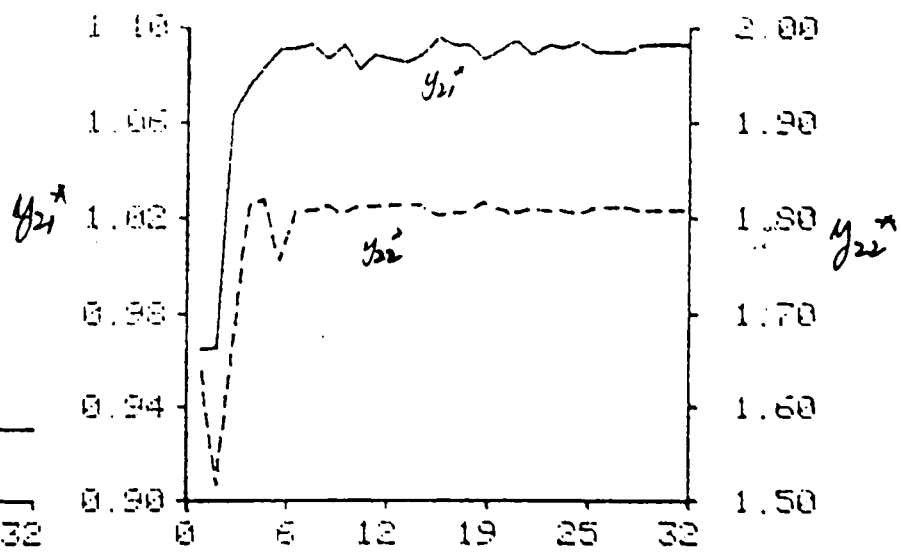
c) LDU1 controls



d) LDU2 controls

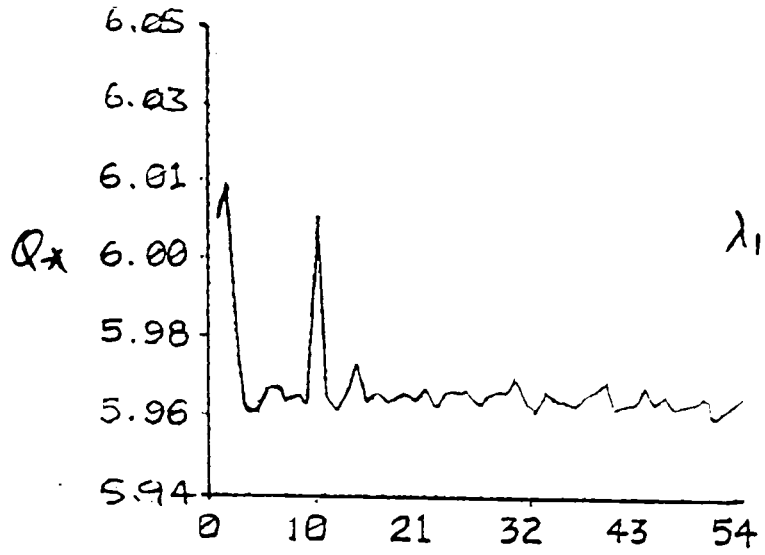


e) LDU1 real output

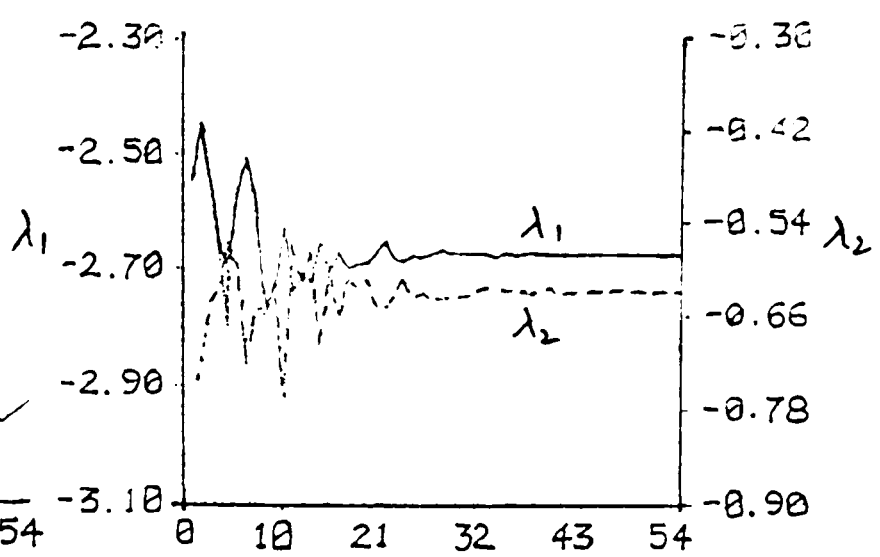


f) LDU2 real outputs

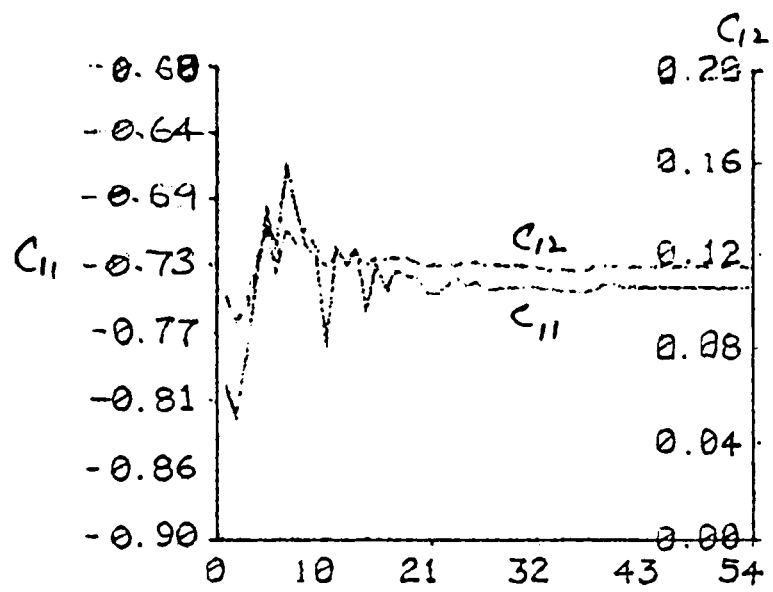
Fig.7.3.1.3 Asynchronous IPMLF simulation result: with open-loop solution and zero shift vector as the starting point and LDU2 delayed by 5 seconds.



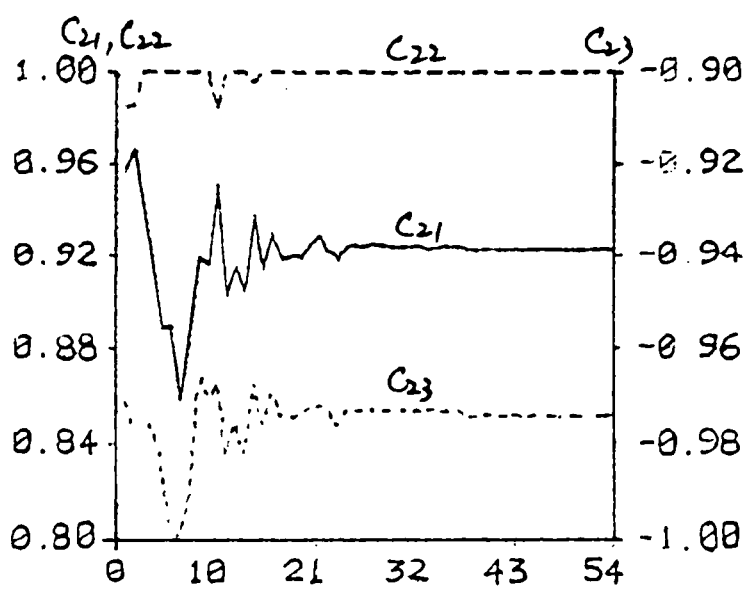
a) Real performance index



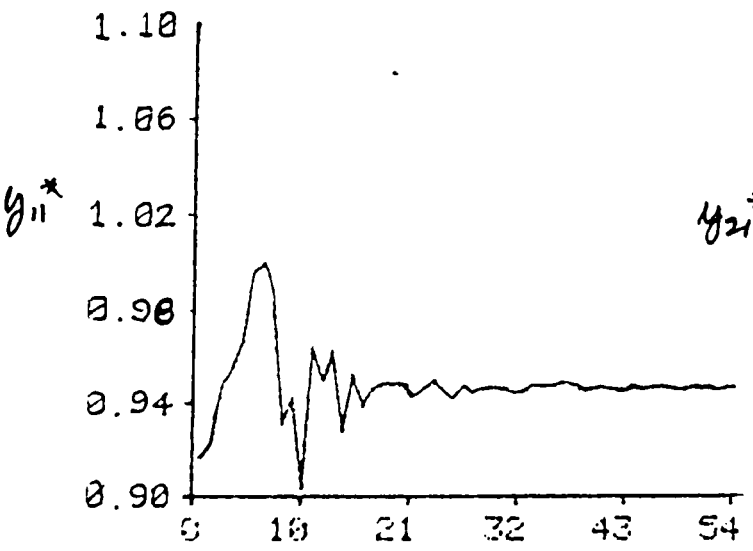
b) Lagrange multipliers



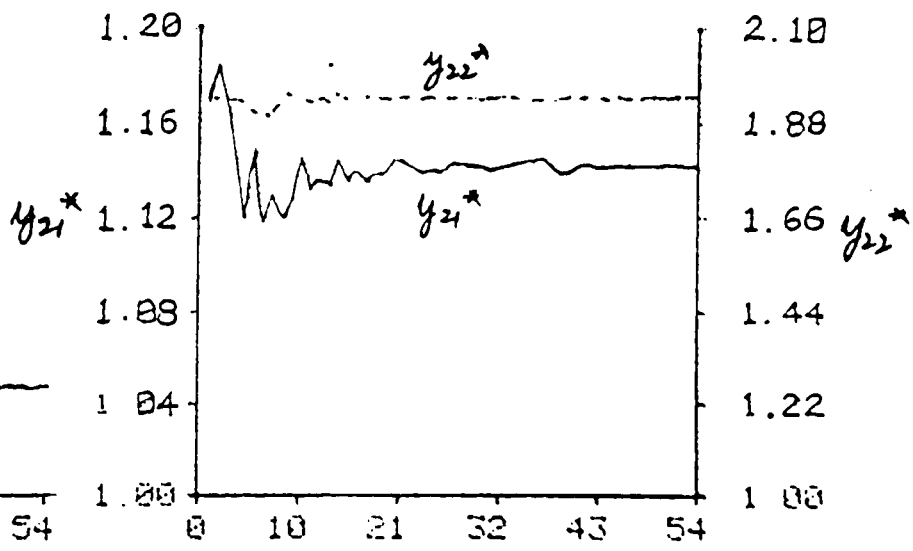
c) LDU1 controls



d) LDU2 controls

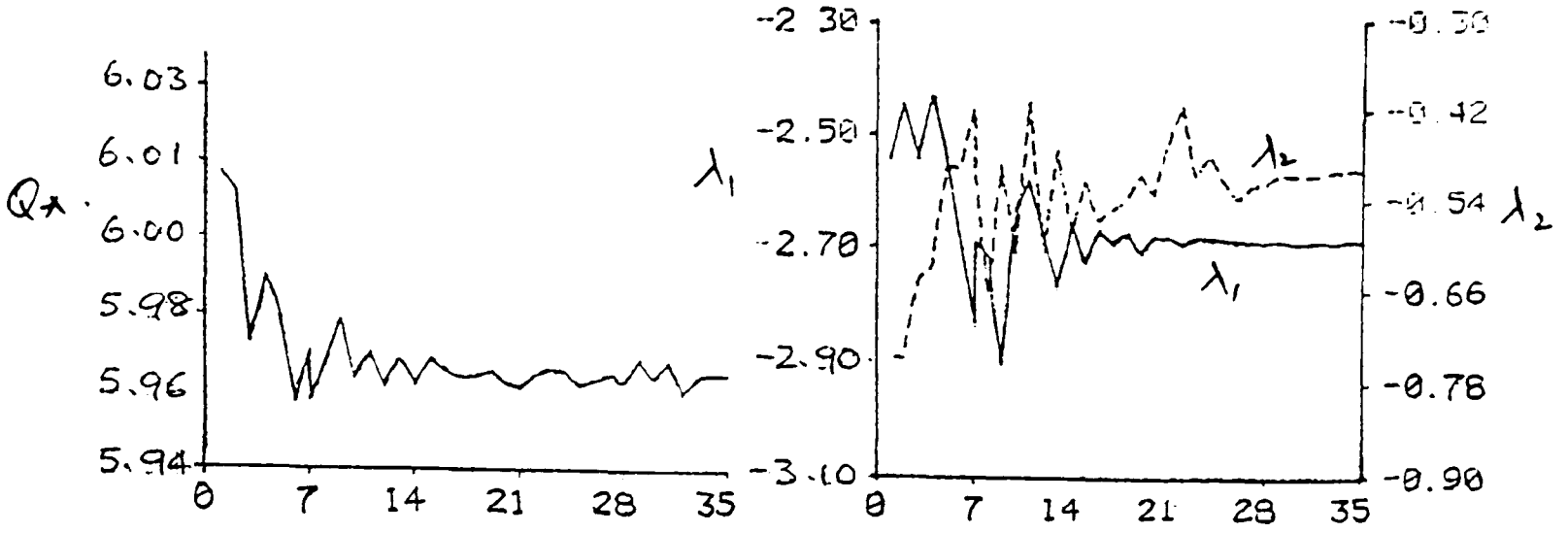


e) LDU1 real output

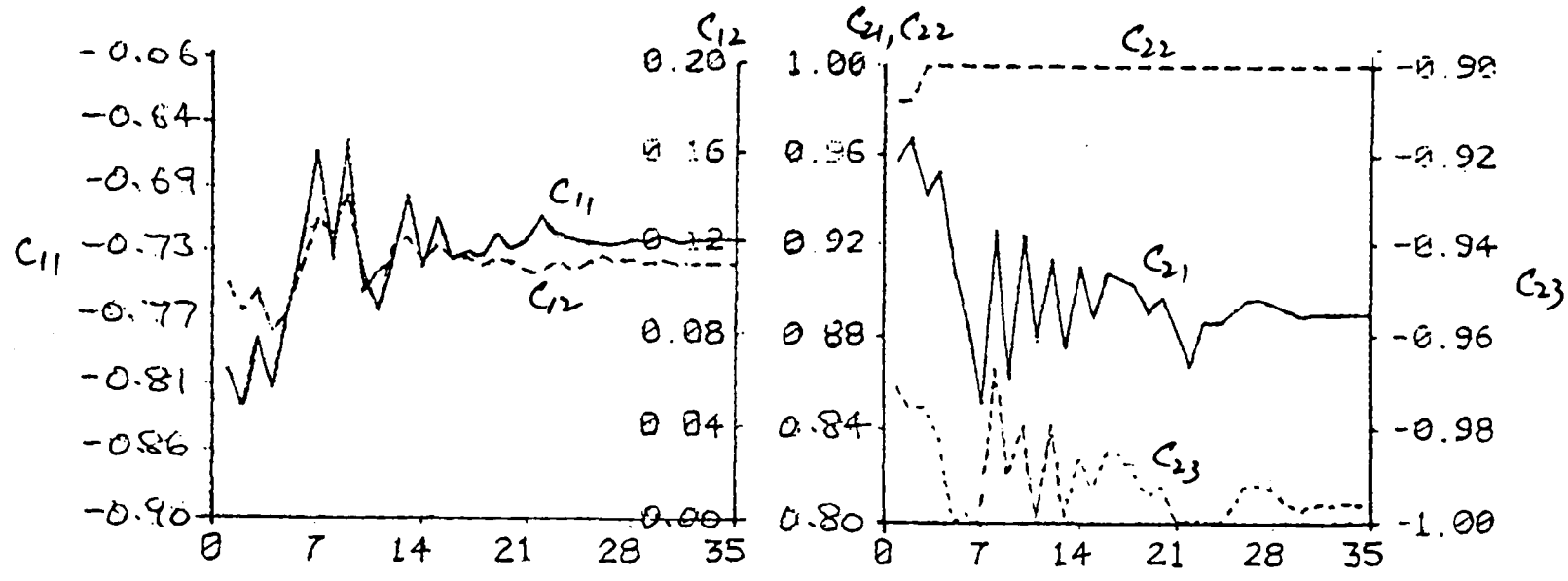


f) LDU2 real outputs

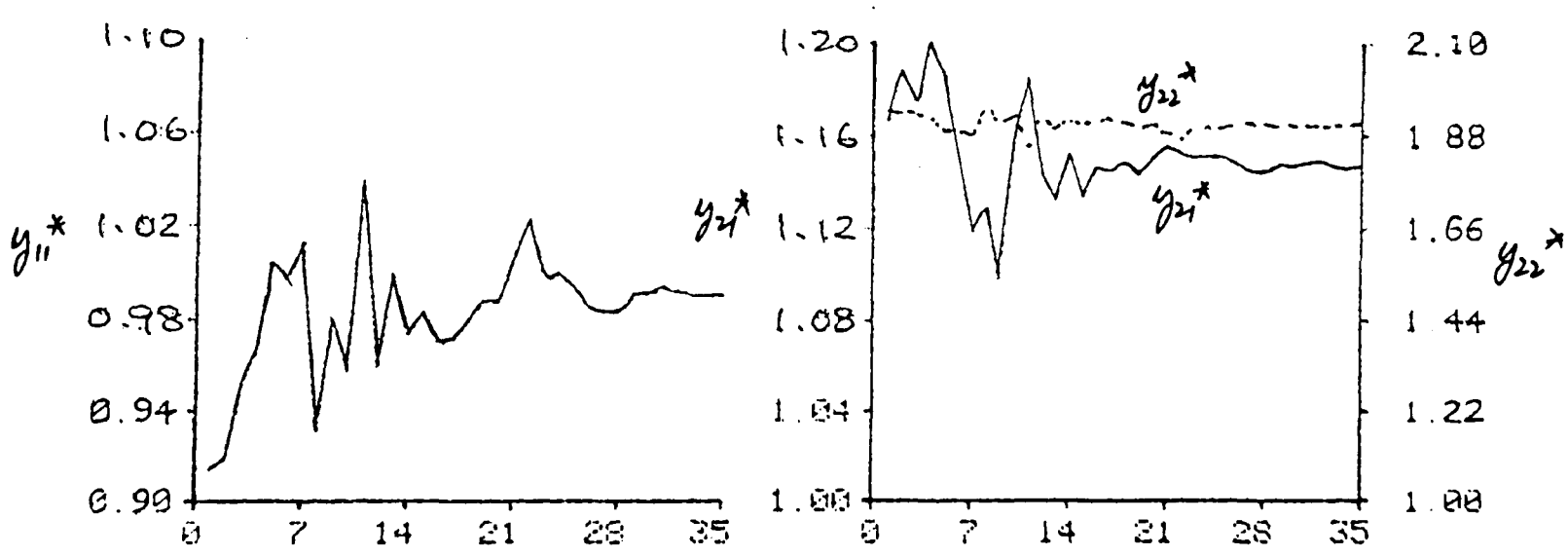
Fig.7.3.2.1. Asynchronous IBMLF simulation result: with T_W = 9 sec, $\underline{K}$ = (0.65; 0.80) and open-loop solution as the starting point.



a) Real performance index b) Lagrange multipliers



c) LDU1 controls d) LDU2 controls



e) LDU1 real output f) LDU2 real outputs

Fig.7.3.2.2. Asynchronous IBMLF simulation result: new local iteration updating scheme with T_W = 5 sec, $\underline{K}$ = (0.65; 0.80) and open-loop solution as the starting point.

CHAPTER 8 CONCLUSIONS

Using the Interaction Balance coordination method, two optimisation algorithms, namely, the Steepest Descent and Constrained Simplex algorithms had been used to solve the coordinator optimisation problem for global and local feedback respectively. The fact that two different optimisation algorithms had been used necessitates some considerations to be taken when comparing the convergence properties of the two mentioned feedback coordination methods. Using local feedback scheme, the infimal level required an average of eight iterations before converging to its solution. This implied that local feedback, on average, required eight times more on-line computation time for the overall optimisation problem than using global feedback. However, in this particular example, better accuracy was obtained using local feedback. On the other hand, local feedback was not as robust as global feedback because of the stability problems occuring in the local decision problems (Roberts,1983).

Constrained Simplex method had been used to solve the IPM coordinator problem. The Interaction Prediction method with global feedback gave the best performance (5.933) among the simulation results obtained in this particular example. With local feedback, the system performance obtained was worse compared with the global feedback. However, the coordinator required less iterations to converge to its optimal solution. In general, this coordination method required solvability and feasibility analysis for constrained optimisation problems which might be very difficult in many practical cases.

With global feedback, tests had been performed to investigate the effects of disturbances, measurement errors and the integrity of the overall system subjected to subprocess and decision unit failures. Disturbances to the real subprocesses were introduced to the controls and outputs through function switches and potentiometers in

the analogue computer. The magnitude of the controls and outputs could be varied from a rated value to zero. Both the IBMGF and IPMGF were quite indifferent to the disturbances imposed on the controls. i.e. the presence of such disturbances would move the performance index to a new value but will return to its original undisturbed value once the disturbances were removed. The system remained stable during the presence of disturbances. Measurement errors were introduced to the interconnection variables using potentiometers with coefficient ranging from unity to zero. When measurement errors of interaction variables (i.e. u_*) or interconnection failure were concerned, IPMGF was quite indifferent. IBMGF was very sensitive to the measurement error in the interconnection inputs. For this particular example, with the potentiometer coefficient denoted by "alpha", the system became unstable if alpha < 0.75, oscillated if 0.75 < alpha < 0.94, and converged to the final solution if alpha $\geq$ 0.95.

The integrity of the overall system subjected to subprocess and decision units failures had been investigated. The subprocess and decision units (LDU) failures were simulated by disconnecting the appropriate controls within the analogue computer using function switches. Both IBMGF and IPMLF were disturbed by switching off the controls corresponding to (LDU1), (LDU2) and (LDU1 & LDU2). The system (coordinator) managed to iterate to a new solution and returned to its original solution once the controls were re-connected. The response of the overall system moved to a new solution and returned to its original solution depending on the failure of which decision unit. Simulation tests showed that the response time of the disturbed system subjected to the failure of the decision unit increased in the following order: (LDU2), (LDU1), (LDU1 & LDU2).

Convergency and stability of local decision units operating asynchronously (i.e. the decision units were not synchronised before applying controls to the real

subprocess) at the infimal level had been investigated. When coordinated by IPMLF, the decision units were stable and converged to their optimal solution without synchronisation. The rate of convergence was quite fast and on an average required 29% more computation to converge to the final solution than the synchronised iterative scheme. However, the decision units became unstable when the system was coordinated by IBMLF. Stabilisation of the decision units under asynchronous operation had been achieved by choosing a suitable loop gain of the local decision problem and introducing a waiting time in the decision units before sending the controls to the simulated subprocesses. Convergency in the decision unit level was comparatively slow which accounted for about 50% more computation than the system with synchronisation. The instability of IBMLF under asynchronous operation was understandable because disturbances would be induced to the interconnection variables. It had already been discovered during the course of investigation of IBMGF that the IBM coordination method was very sensitive to disturbances imposed on the interconnection variables.

In this research, investigation of closed-loop hierarchical control and optimisation of an interconnected process simulated by an analogue computer had been performed. Both on-line coordination methods, namely, the Interaction Balance Method and the Interaction Prediction Method with local or global feedback had been implemented successfully using the distributed hierarchical computer system. The effect of asynchronised iteration of local decision units upon the stability and convergency of local decision units, and the coordinator had also been studied. This research is mainly experimental in nature. In order to guarantee stability in the coordinator and local decision optimisation problems, future work needs to be done on a theoretical basis to investigate local decision non-synchronisation aspects and associated convergence properties.

Since the performance of the local decision making process is greatly hindered by the lack of memory and mathematical function supports provided by the I-MICs, therefore, the microcomputers should be upgraded in order to implement or develop advanced parameter estimation techniques for asynchronised local decision iteration.

Furthermore, investigation of the application of other on-line coordination methods, e.g. "Mixed Method" - a combination of Interaction Balance Method and Interaction Prediction Method, for hierarchical control and optimisation of large scale systems should be performed.

Finally, the on-line coordination methods should be applied to control real systems, e.g. pilot scale industrial processes other than the simulated interconnected process.

REFERENCES AND BIBLIOGRAPHY

REFERENCES

Bertsekas D.P. (1982)
Constrained Optimization and Lagrange Multiplier Methods. Academic Press.

Brdys M. and B. Ulanicki (1978)
On the Completely Decentralised Control with Local Feedback in Large Scale System. Arch. Automat. i Telemech., 21-35.

Brdys M. and P. Michalak (1978)
On-Line Coordination with Local Feedback for Steady State Systems. Arch. Automat. i Telemech., 404-421.

Brdys M., P. Michalak and B. Ulanicki (1982)
Optimising Control of Large Scale Systems under Time-Varying Disturbances by Price Mechanism with Local Feedback. Large Scale Systems 3, 123-142.

Brdys M. (1983)
Hierarchical Optimising Control of Steady State Large Scale Systems under Model-Reality Differences of Mixed Type - A Mutually Interacting Approach. IFAC/IFORS, LSSTA, 49-57.

Findeisen W., F.N. Bailey, M. Brdys, K. Malinowski, P. Tatjewski and A. Wozniak (1980)
Control and Coordination in Hierarchical Systems, IIASA, J. Wiley.

Gyles V.B. (1983)
Design of Real Time Estimation Algorithms for Implementation in Microprocessor and Distributed Processor Systems. In "Control and Dynamic Systems, Advances in Theory and Applications" by C.T. Leondes, Vol.19, Academic Press.

Luenberger D.G. (1984)
Linear and Nonlinear Programming, 2nd Edition, Addison-Wesley.

Malinowski K. and A. Ruszczynski (1975)
Application of Interaction Balance Method to Real Process Coordination. Control & Cybernetics, Vol.4, No.2.

Malinowski K. (1983)
Practical Aspects of Coordination Processes. IFAC/IFORS, LSSTA, 309-317.

Mesarovic M.D., D. Macko and Y. Takahara (1970)
Theory of Hierarchical, Multilevel, Systems. Academic Press.

NAG
E04CCF - Simplex Method

Roberts P.D., J.E. Ellis, C.W. Li, F.Q. Shao, P. Zheng and B.W. Wan (1983)
Algorithms for Hierarchical Control Steady State Optimisation of Industrial Processes. IFAC/IFORS, LSSTA, 425-431.

Roberts P.D., C.W. Li, I.A. Stevenson and D.S. Wadhwani (1984)
On-line Distributed Hierarchical Control and Optimisation of Large Scale Processes Using a Micro-computer Based System. First European Workshop on 'Real-time Control of Large Scale Systems', Patras, Greece, 432-441.

Shao F.Q. and P.D. Roberts (1983)
A Price Correction Mechanism with Global Feedback for Hierarchical Control of Steady State Systems. Large Scale Systems 4, 67-80.

Shao F.Q. and P.D. Roberts (1983)
Augmented Decentralised Control with Local Feedback for Steady State Systems. Research Memo., DSS., TCU.

Stevenson I.A. (1982)
Computer Control of a Travelling Load Furnace M.Sc Thesis, TCU.

Stevenson I.A. (1984)
Recent Improvements to the Distributed Computer Control System and a Survey of On-Line Discrete-Time Parameter Estimation Algorithms for Use on the Travelling-Load Furnace. Research Memo., CEC/IAS 5, TCU.

Szymanowski J. and A. Ruszczynski (1979)
Convergence Analysis for Two Level Algorithms of Mathematical Programming. Mathematical Programming Study 10, 158-171.

Tatjewski P. and M. Cygler (1981)
Completely Decentralised Output Control Based on an Approximate Mathematical Model. Large Scale Systems 2, 243-255.

Tatjewski P. (1983)
On-Line Steady State Control of Large Scale Non-linear Interconnected Systems Using Augmented Interaction Balance Method with Feedback. IFAC/IFORS, LSSTA, 526-531.

Wan B.W. and P.D. Roberts (1981)
Some Improvements in Techniques for Hierarchical Control of Steady State Systems. Research Memo., DSS/B-WW-PDR/229, TCU.

Wismer D.A. (1971)
Distributed Multilevel Systems. In "Optimisation Methods for Large Scale Systems", 233-273. Mcgraw Hill.

Roberts P.D. (1982)
Lecture notes on Large Scale Systems, MSc course on Systems Engineering, City University, Department of Electrical, Electronic and Information Engineering.

BIBLIOGRAPOHY

Baudet G.M. (1978)
Asynchronous Iterative Methods for Microprocessors. J. of Assoc. for Computing Machinery, Vol.25, No.2, 226-244.

Bernussou J. and J.C. Geromel (1979)
Stability Approach to Robust Control for Interconnected Systems. In 'Large Scale Systems Engineering Applications'. ed. by M.G. Singh, North-Holland, 66-78.

Bertsekas D.P. (1976)
Multiplier Methods: A Survey. Automatica, Vol.12, 133-145.

Bertsekas D.P. (1983)
Distributed Asynchronous Computation of Fixed Points. Mathematical Programming 27, 107-120.

Bertsekas D.P. (1984)
Convergence Analysis of Distributed Asynchronous Iterative Processes. IFAC 9th World Congress, Vol.X, Colloquia 11.1/b-2, 188-189.

Box M.J. (1966)
A Comparison of Several Current Optimisation Methods, and the Use of Transformations in Constrained Problems. The Computer Journals, Vol.9, 67-77.

Brdys M. and E.B. Lee (1982)
Completely Decentralised Stabilisation of Complex Processes with Delayed Interactions. Inter. Wiss. Koll. Th. Ilmenau, 123-126.

Chazan D. and W. Miranker (1969)
Chaotic Relaxation. Linear algebra and its Applications, 2, 199-222.

Cohen G. (1980)
Auxilary Problem Principle and Decomposition of Optimisation Problems. J. of Optim. Theory and Applic., Vol.32, No.3, 277-305.

Fletcher R. and M.J.D. Powell (1963)
A Rapid Convergent Descent Method for Minimisation. The Computer Journal, Vol.6, 163-168.

Hager W.W. (1979)
Lipschitz continuity for Constrained Processes. SIAM J. Control & Optimisation, Vol.17, No.3, 321-338.

Jamshidi M. (1983)
Large Scale Systems: Modelling and Control. North-Holland.

Kung H.T. (1976)
Synchronous and Asynchronous Parallel Algorithms for Multiprocessors. In 'Algorithms and Complexity: New Directions and Recent Results', ed. by J.F. Traub, Academic Press, New York, 153-199.

Lefkowitz I. and M.R. Buchner (1982)
Distributed Control: Status and Opportunities. IFAC, CIDCS, 1-12.

Ruszczynski A. (1976)
Convergence Conditions for the Interaction Balance Algorithm Based on an Approximate Mathematical Model. Control and Cybernetics, Vol.5, No.4, 29-42.

Sandell N.R., P. Varaiya, M. Athans and M.G. Safonov (1978)
Survey of Decentralized Control Methods for Large Scale Systems.
IEEE Trans. on Automatic Control, Vol.AC-23, No.2, 108-128.

Singh M.G. and A. Titli (1978)
Systems - Decomposition, Optimisation and Control. Pergamon.

Singh M.G., K. Malinowski, M.S. Mahmoud and A.Titli (1982)
Hierarchical Optimisation and Control - A Survey. Control Systems Centre, Report No.540, U.M.I.S.T.

Suh I.H. and Z. Bien (1982)
A note on the Stability of Large Scale Systems with Delays. IEEE Trans. of Auto. Control, AC-27, No.1, 256-258.

Tsitsiklis J.N., D.P. Bertsekas and M. Athans (1986)
Distributed Asynchronous Deterministic and Stochastic Gradient Optimization Algorithms.
IEEE Trans. on Auto. Control, Vol.AC-31,No.9, 803-812.

Appendix A: Local Decision Feasibility Set under Constraints

Due to the lack of programming memory, mathematical functions and subroutines available in the micro-computers (local decision units), standard library numerical optimisation algorithms could not be used to solve the local decision problems which were constrained optimisation problems with equality and inequality constraints. Hence, each local optimisation problem had to be solved analytically using the idea of 'active set method' and the theory of extrema.

The 'active set method' is an algorithm for solving optimisation problem with inequality constraints. The concept underlying this method is to consider the inequality constraints which consist of two parts: those that are to be treated as active and those that are to be treated as inactive. The inactive constraints are essentially ignored during the optimisation procedure. Therefore if the set of active constraints was known, the original problem could be replaced by the corresponding problem having equality constraints only.

The mechanism of the 'active set method' in solving optimisation problem with inequality constraints is to define, at each iteration step, a set of active constraints, known as the 'working set', which is a subset of the constraint set of the original problem. The current point is therefore feasible for the working set. The algorithm then proceeds to move along the worked surface defined by the active constraints to a new improved point. At this new point, the working set may be changed. The reason for doing so may be due to the addition of constraint(s) to the working set when new constraint(s) are encountered or the dropping of a constraint from the working set to ensure the performance index is strictly decreased.

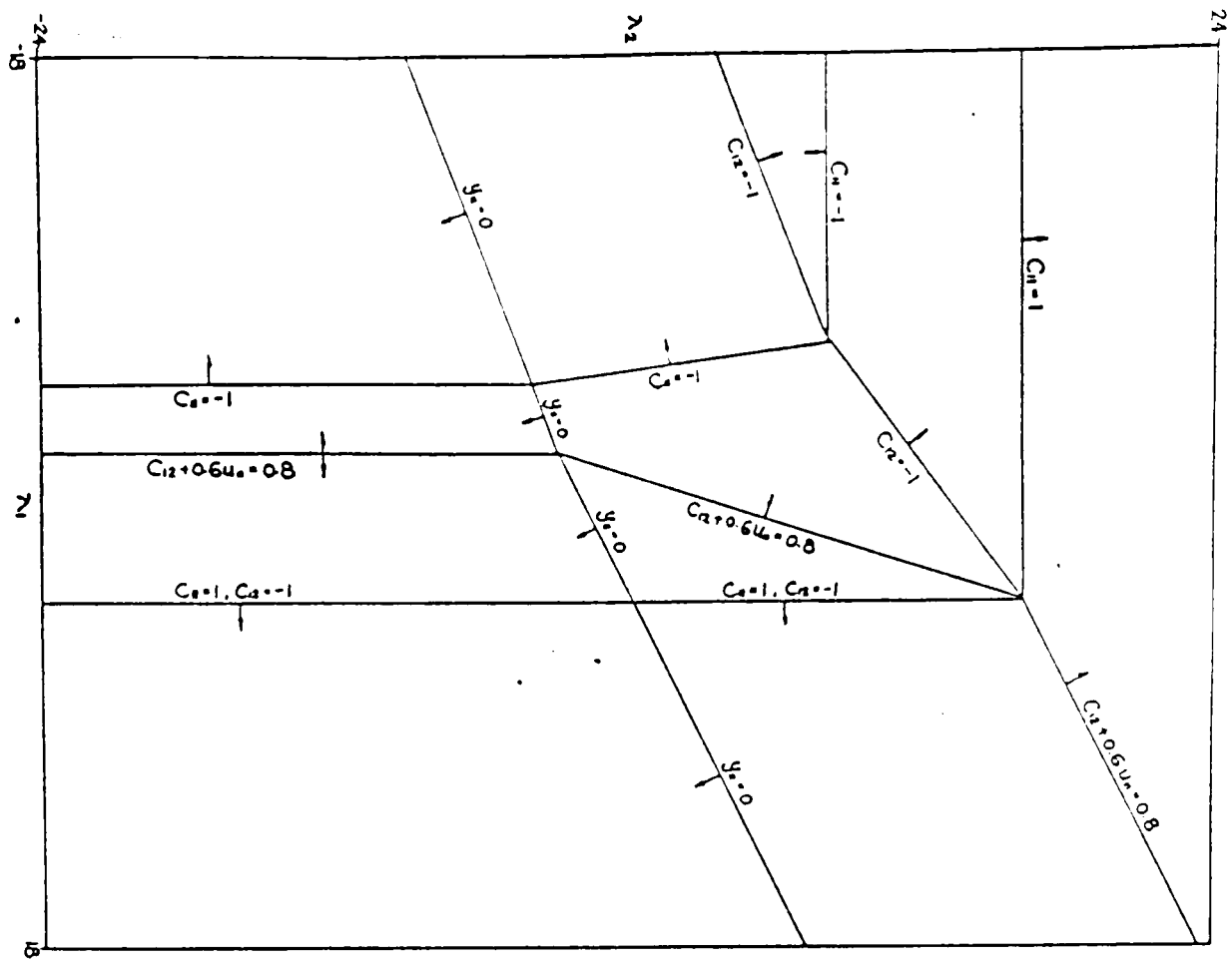
The procedure of determining the feasible solution region of local decision optimisation problem subjected to

equality and inequality constraints is as follows: Starting with the unconstrained solution, the performance index of each local decision problem is differentiated with respect to the manipulated inputs (controls) to obtain the optimal controls in terms of interaction variables. Then these control values are substituted in the constraint sets of the decision unit to search for the active constraints which bound the unconstrained solution. Then the active constraints are plotted with the interaction variables to indicate the bounded unconstrained solution. Once the unconstrained solution region has been found, the active constraints are subsequently used as new constraint boundaries for the generation of new solution regions next to the unconstrained solution. Defining a new solution region by one of the active constraint boundaries previously obtained, the procedure is repeated and a new bounded solution region is found. The complete procedure is then repeated until no further new solution regions are formed.

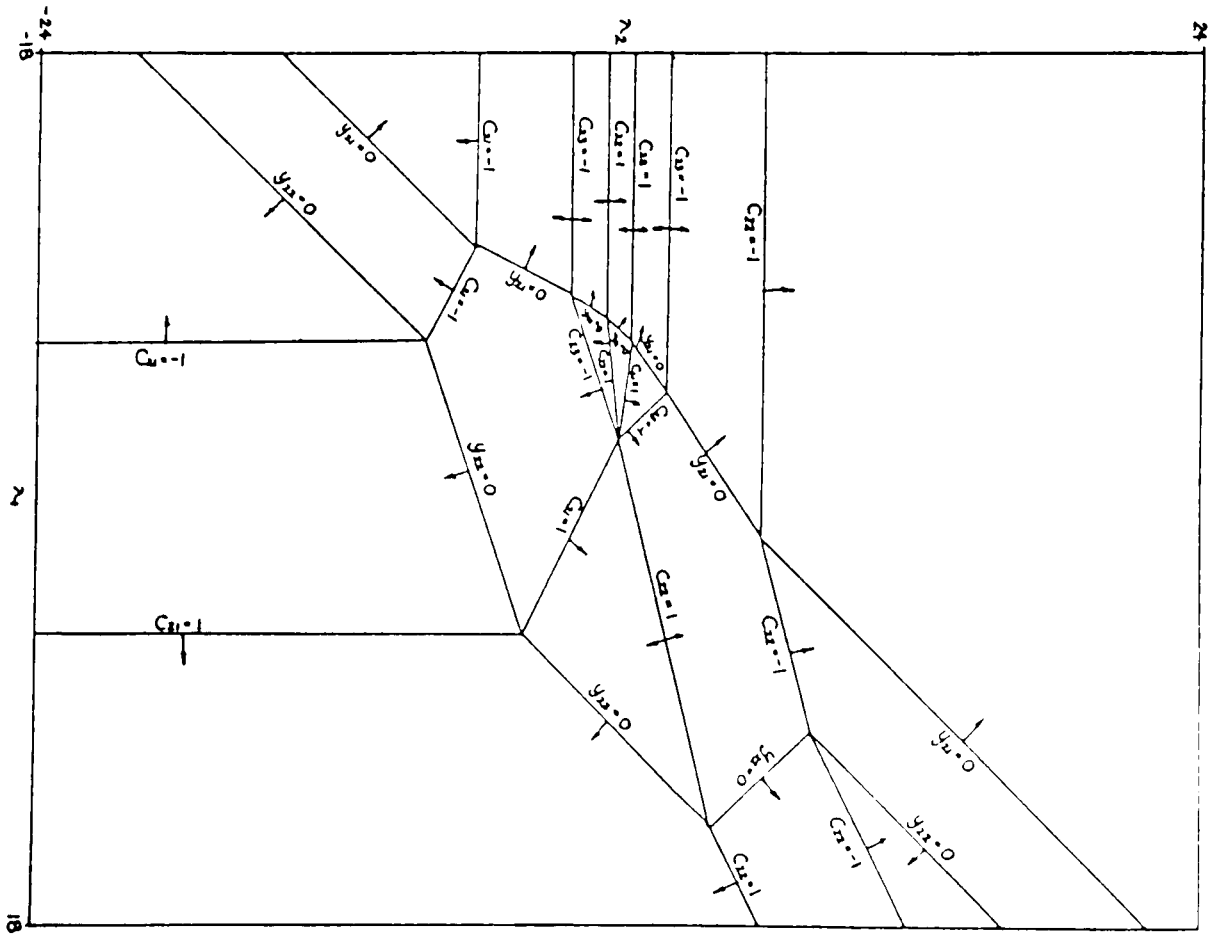
This analytical approach is used to solve the local decision optimisation problems for price coordination strategy. Decision unit solution regions using the interaction balance method with global feedback and local feedback are shown in fig.A1 and fig.A2 respectively.

The drawback of this analytical method for solving the local decision optimisation problem is that it is restricted to simple low dimensional problems because it is impossible to visualise greater than two dimensional solution regions.

Using the 'Active Set Method', decision unit solution regions for the IBMGF, IBMLF and IPMLF will be shown in fig.A1, A2 and A3 respectively.

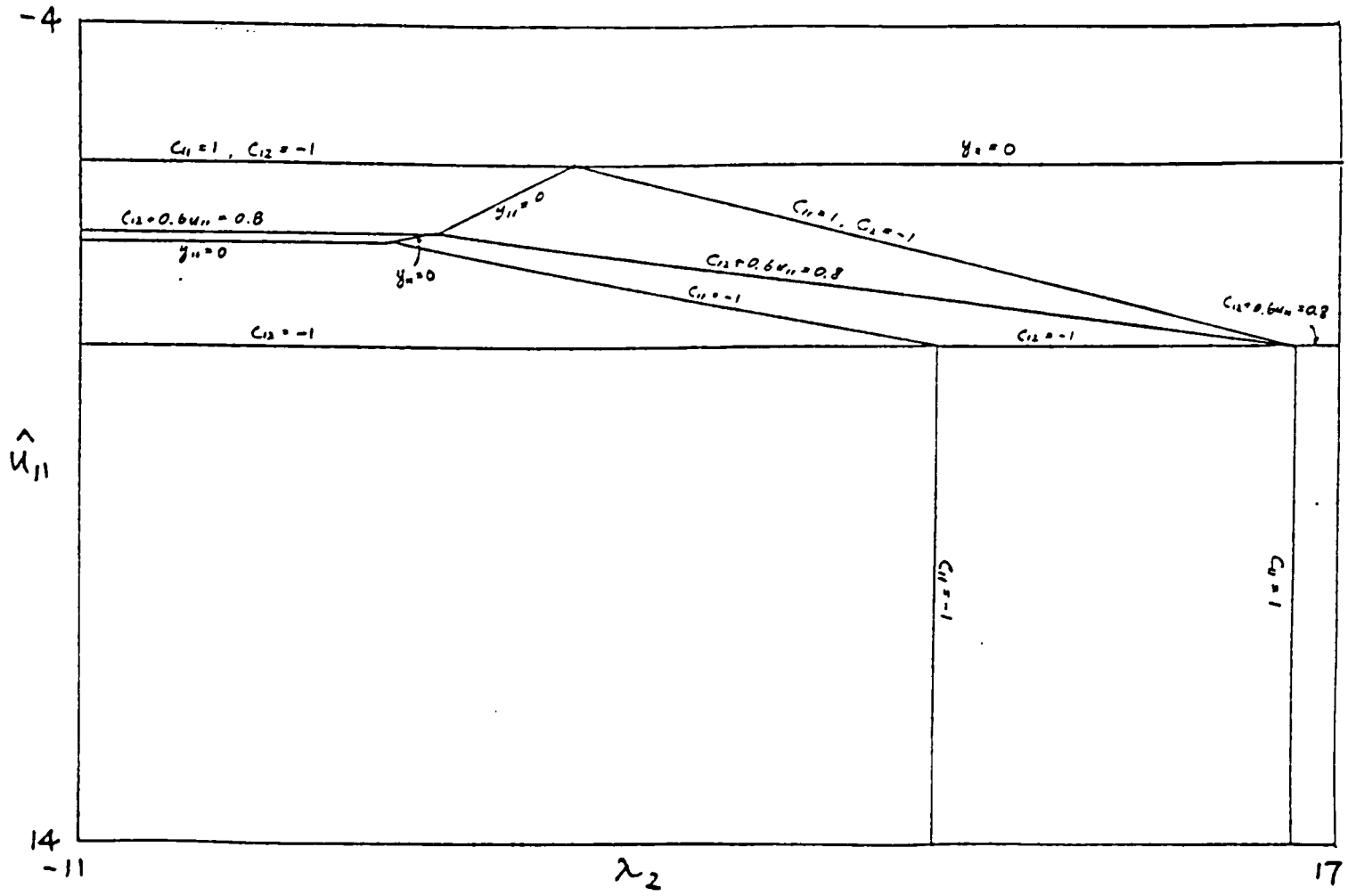


a) Local decision unit 1

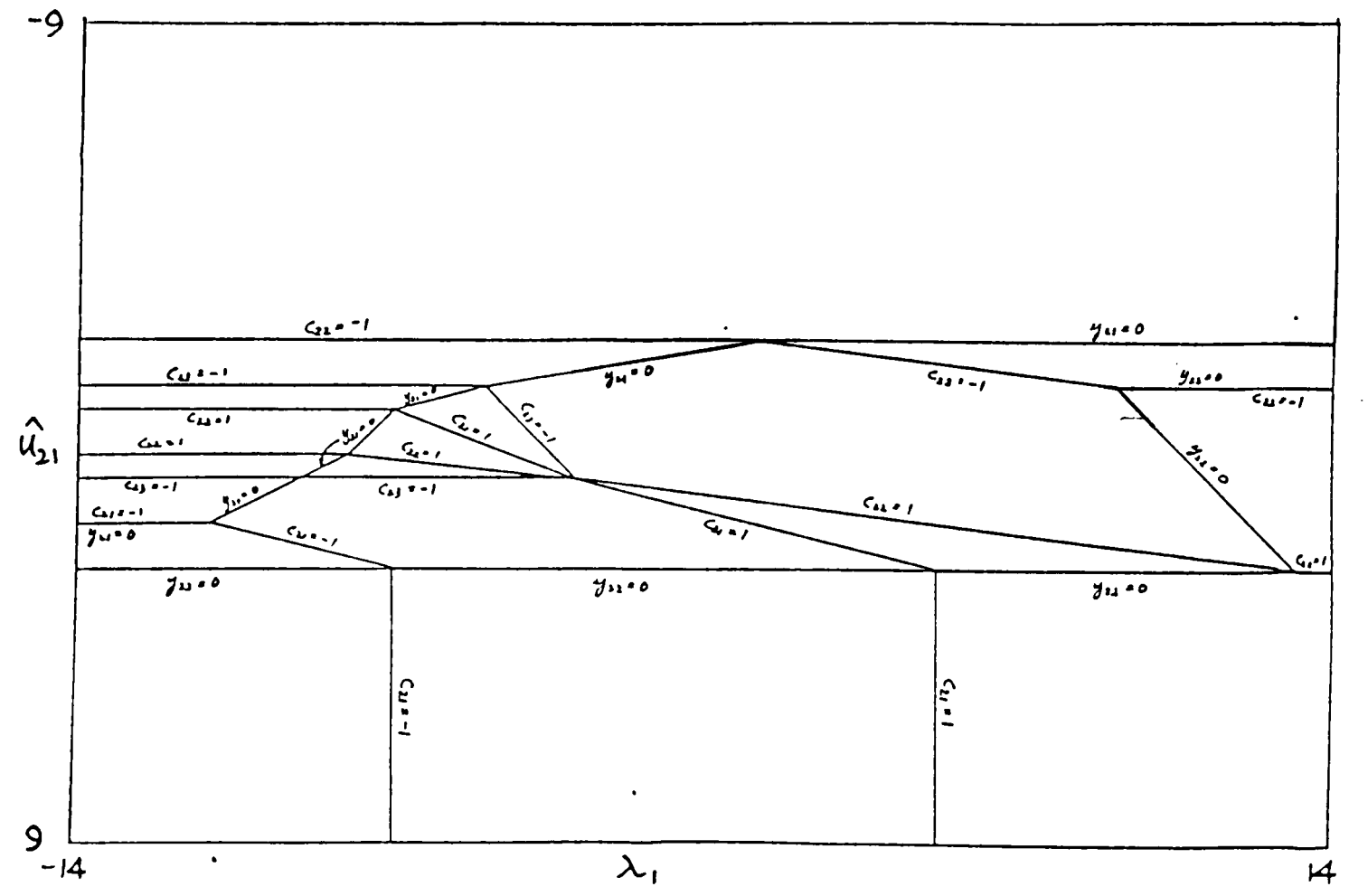


b) Local decision unit 2

Fig.A1 Decision unit solution regions for IBMGF

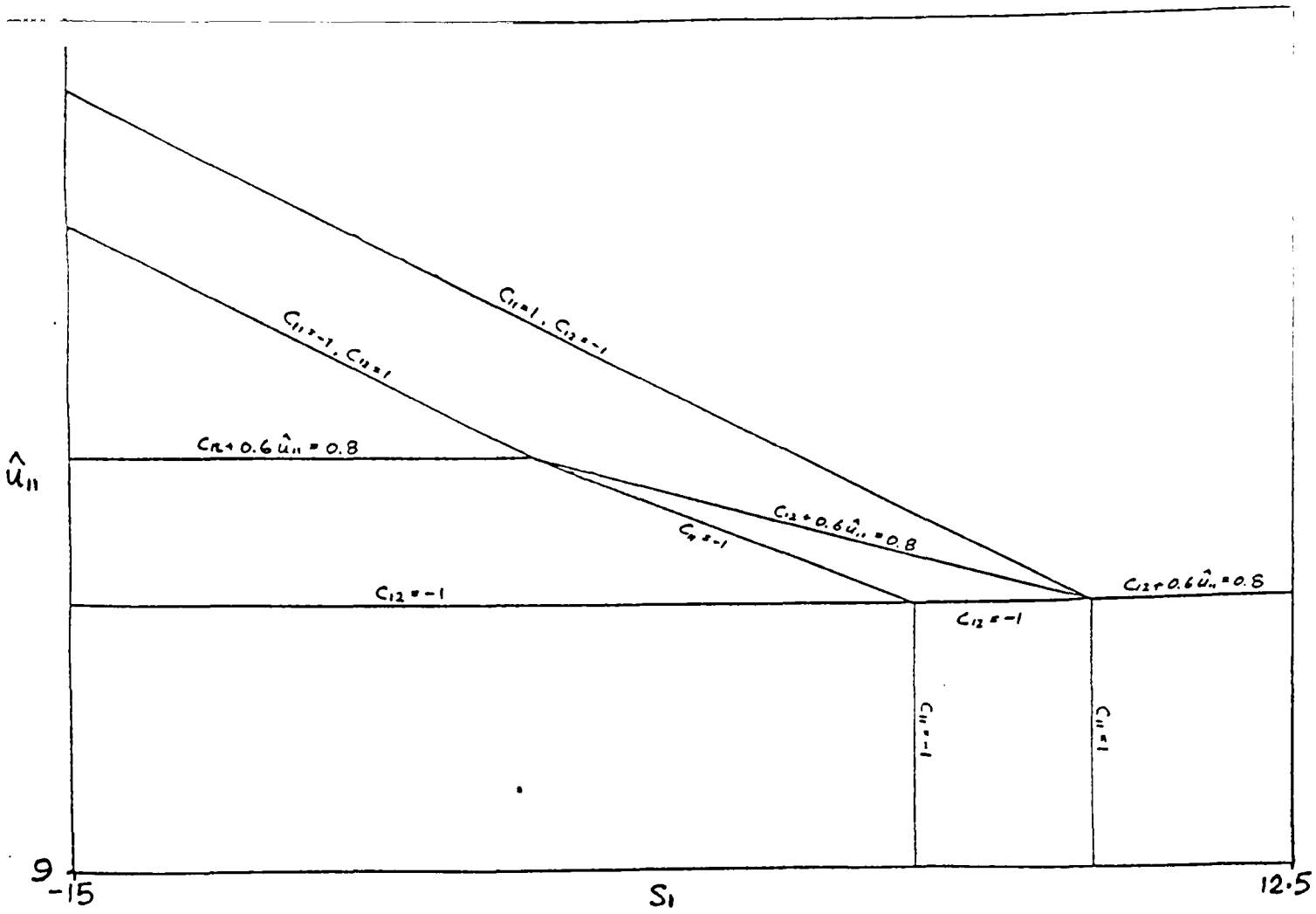


a) Local decision unit 1

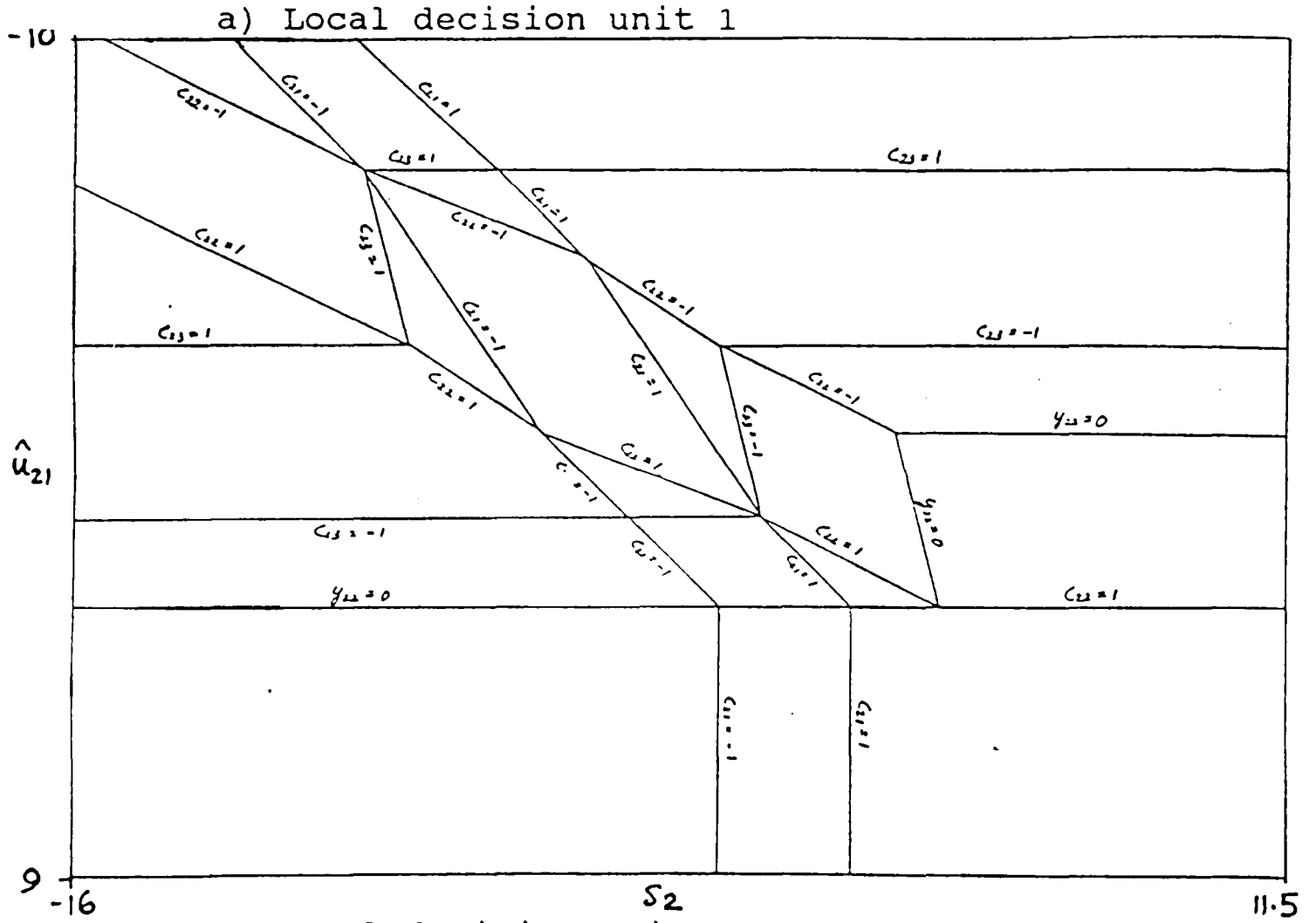


b) Local decision unit 2

Fig.A2 Decision unit solution regions for IBMLF



a) Local decision unit 1



b) Local decision unit 2

Fig.A3 Decision unit solution regions for IPMLF

B1. Coordinator Software Listing

Foreground Routines

```
      SUBROUTINE LNKSET
      INTEGER*2 SIZE(2),SECS,MBUFS(1000,2),LAST1(2),NEXT1(2)
      LOGICAL*1 MBFREE(2),RXFREE(2)
      BYTE ERROR(2)
      COMMON/FLAGS/RXFREE
      COMMON/MBUFS/MBUFS,MBFREE,LAST1,NEXT1,SIZE,SECS
C
C TO INITIALISE I-MIC LINKS.
C
C
      SECS=1
      DO 1000  I=1,2
      NEXT1(I)=1
      LAST1(I)=-1
      SIZE(I)=500
      RXFREE(I)=.TRUE.
      ERROR(I)=0
1000  MBFREE(I)=.TRUE.
C
      CALL INITD(2,3)
      TYPE 7
7     FORMAT(1H ,' Link nos. 2 & 3 initialised.')
      CALL RK(0)
      RETURN
      END

      SUBROUTINE DATEO
C
C    THIS SUBROUTINE PRINTS THE CURRENT DATE
C    IN DAY,MONTH,YEAR
C
      LOGICAL*1 A(9)
C
      CALL DATE(A)
      PRINT 1,A
1     FORMAT(' DATE: ',9A1)
      RETURN
      END

      SUBROUTINE TIME
C
C    THIS SUBROUTINE PRINTS THE CURRENT TIME OF THE DAY
C    IN HRS,MINS,SECS.
C
      INTEGER*2 ITIME,IHRS,IMIN,ISEC,ITCK
C
C
      CALL GTIM(ITIME)
      CALL CVTTIM(ITIME,IHRS,IMIN,ISEC,ITCK)
      PRINT 1,IHRS,IMIN,ISEC
1     FORMAT(' TIME: ',I2,':',I2,':',I2,,//)
      RETURN
      END
```

```
C
      SUBROUTINE INITD(I,J)
C
C     THIS ROUTINE CALCULATES THE ADDRESSES & VECTORS FOR THE
C     SPECIFIED LINKS AND CONNECTS THE INTERRUPT HANDLERS 'RXIHA2'
C     'RXIHA3','TXIHA2','TXIHA3'.
C
      BYTE FLAG1(2),FLAG11(2),FLAG21(2),FLAG31(2)
      INTEGER*2 IVEC(2),RCSR(2),RBUF(2),XBUF(2),XCSR(2),BIN(2),BOUT(2)
      COMMON/FLAGS/FLAG1,FLAG11,FLAG31,FLAG21
      COMMON/COUNT/BIN,BOUT
      COMMON/LCHAR/IVEC,RCSR,RBUF,XBUF,XCSR
      EXTERNAL TXIHA2,TXIHA3,RXIHA2,RXIHA3
C
C                                     CALCULATES VECTORS FOR SPECIFIED
C
      IF(I.LE.2)IVEC(1)='350+'10*I
      IF(I.GT.2)IVEC(1)='300+'10*I
      IF(J.LE.2)IVEC(2)='350+'10*J
      IF(J.GT.2)IVEC(2)='300+'10*J
C
C                                     AND THE ADDRESSES
C
      RCSR(1)='176470+'10*I
      RCSR(2)='176470+'10*J
      RBUF(1)=RCSR(1)+2
      RBUF(2)=RCSR(2)+2
      XBUF(1)=RCSR(1)+6
      XBUF(2)=RCSR(2)+6
      XCSR(1)=RCSR(1)+4
      XCSR(2)=RCSR(2)+4
C
C                                     CONNECT RX INTERRUPT HANDLERS
C
      I=ICNINT(IVEC(1),RXIHA2,7)
      IF(I.GE.0)GOTO 10
      I=ICNINT(IVEC(2),RXIHA3,7)
      IF(I.GE.0)GOTO 10
C
C                                     CONNECT TX INTERRUPT HANDLERS
C
      I=ICNINT(IVEC(1)+4,TXIHA2,7)
      IF(I.GE.0)GOTO 20
      I=ICNINT(IVEC(2)+4,TXIHA3,7)
      IF(I.GE.0)GOTO 20
      DO 30 I=1,2
      BIN(I)=0
      BOUT(I)=-2
      FLAG1(I)=1
      FLAG11(I)=1
      FLAG21(I)=0
      FLAG31(I)=0
30    CONTINUE
C
```

```
C                                               MAP THE I/O PAGE INTO THE PROGRAM
C
      CALL MFIOPS
C
C                                               ENABLE RX INTERRUPTS
C
      I=IPEEK(RCSR(1))
      CALL IPOKE(RCSR(1),'100.OR.I)
      I=IPEEK(RCSR(2))
      CALL IPOKE(RCSR(2),'100.OR.I)
      RETURN
10    WRITE(5,100)
      STOP
20    WRITE(5,110)
      STOP
100   FORMAT(1H ,'UNABLE TO CONNECT RX INTERRUPT VECTOR')
110   FORMAT(1H ,'UNABLE TO CONNECT TX INTERRUPT VECTOR')
      END
C
C
C
C
C
C
      SUBROUTINE RXIPRO(ID)
      INTEGER*2 IVEC(2),RCSR(2),RBUF(2),XBUF(2),XCSR(2),BIN(2),BOUT(2)
      INTEGER*2 TEMP,EXPI
      BYTE FLAG1(2),FLAG11(2),FLAG21(2),FLAG31(2),RXBUF(256,2),TEMPB(2)
      COMMON/LCHAR/IVEC,RCSR,RBUF,XBUF,XCSR
      COMMON/FLAGS/FLAG1,FLAG11,FLAG31,FLAG21
      COMMON/COUNT/BIN,BOUT
      COMMON/RX/RXBUF
      EQUIVALENCE (TEMP,TEMPB(1))
C
C                                               CANCEL ITIMER REQUEST (IF ANY)
C
      CALL ICMRT(1,EXPI)
C
C
C                                               LOCK HANDLER IN MEMORY
C
      CALL LKANMY
      CALL MFIOPS
C
C                                               READ  A BYTE FROM THE LINK
C
      TEMP=IPEEK(RBUF(ID))
      IF((TEMP.LT.0).OR.(FLAG31(ID).NE.0)) GOTO 4
      IF(FLAG21(ID).EQ.0)GOTO 3
C
C                                               ENABLE TX INTERRUPTS BEFORE EXIT
C
4     I=IPEEK(XCSR(ID))
      CALL IPOKE(XCSR(ID),'100.OR.I)
      RETURN
3     IF((BIN(ID).NE.0).OR.(FLAG1(ID).NE.0))GOTO 1
```

```
C
C                                          FLAG31 IS THE LINK ERROR FLAG
C
      FLAG31(ID)=1
      RETURN
1     RXBUF(BIN(ID)+1,ID)=TEMPB(1)
      BIN(ID)=BIN(ID)+1
      IF(BIN(ID).LE.RXBUF(1,ID)) GOTO 4
      FLAG1(ID)=0
      BIN(ID)=0
C
C                                          FLAG21 SET AT END OF RECEIVE
C
      FLAG21(ID)=1
      GOTO 4
      END
C
C
C
C
C
C
      SUBROUTINE TXIFRO(ID)
      INTEGER*2 IVEC(2),RCSR(2),RBUF(2),XBUF(2),XCSR(2),BIN(2),BOUT(2),TEMF
      BYTE FLAG1(2),FLAG11(2),FLAG21(2),FLAG31(2),TXBUF(256,2),TEMPB(2)
      COMMON/LCHAR/IVEC,RCSR,RBUF,XBUF,XCSR
      COMMON/FLAGS/FLAG1,FLAG11,FLAG31,FLAG21
      COMMON/COUNT/BIN,BOUT
      COMMON/TX/TXBUF
      EQUIVALENCE (TEMF,TEMPB(1))
      CALL LKANMY
      CALL MFIOPS
C
C                                          DISABLE TX INTERRUPTS
C
      I=IPEEK(XCSR(ID))
      CALL IPOKE(XCSR(ID),'177677.AND.I)
C
C                                          DELAY TO ALLOW I-MIC TO CATCH UP
C
      DO 10 J=1,250
10    CONTINUE
      IF(FLAG21(ID).EQ.0) GOTO 5
      IF(BOUT(ID).EQ.-2)GOTO 6
      TEMPB(2)=0
      IF(BOUT(ID).LT.0)GOTO 4
      GOTO 1
C
C                                          FIRST XMITTED BYTE IS FLAG11(TXFREE)
C
4     TEMPB(1)=FLAG11(ID)
      CALL IPOKE(XBUF(ID),TEMF)
      IF(FLAG11(ID).NE.0)GOTO 2
      BOUT(ID)=BOUT(ID)+1
      GOTO 3
1     TEMPB(1)=TXBUF(BOUT(ID)+1,ID)
```

```
C
C                                          XMIT A DATA BYTE
C
      CALL IPOKE(XBUF(ID),TEMP)
      BOUT(ID)=BOUT(ID)+1
      IF(BOUT(ID).LE.TXBUF(1,ID))GOTO 3
      FLAG11(ID)=1
C
C                                          RESET BYTE COUNT & FLAGS
C
2     BOUT(ID)=-2
      FLAG21(ID)=0
C
C                                          CALL COMPLETION ROUTINE (RK(0))
C
      CALL RK(0)
3     RETURN
6     BOUT(ID)=-1
C
C                                          ECHO A '0' WHEN I-MIC IS XMITTING
C
5     CALL IPOKE(XBUF(ID),0)
      RETURN
      END
C
C     RXFORD & TXFORD ARE DATA FORMATTING PROGRAMS TO CONVERT THE
C     BYTE ARRAY RXBUF TO INTEGER*2 AND THE INTEGER*2 ARRAY TX TO BYTE
C
C
C
      SUBROUTINE RXFORD(RX)
      INTEGER*2 RX(128,2),TEMP
      BYTE RXBUF(256,2),TEMPB(2)
      EQUIVALENCE (TEMP,TEMPB(1))
      COMMON/RX/RXBUF
      DO 20 J=1,2
      TEMPB(1)=RXBUF(1,J)
      TEMPB(2)=0
      RX(1,J)=TEMP
      DO 10 I=2,RXBUF(1,J),2
      TEMPB(1)=RXBUF(I,J)
      TEMPB(2)=RXBUF(I+1,J)
10    RX(I/2+1,J)=TEMP
20    CONTINUE
      RETURN
      END
C
C
C
C
C
      SUBROUTINE TXFORD(TX)
      INTEGER*2 TX(128,2),TEMP
      BYTE TXBUF(256,2),TEMPB(2)
      EQUIVALENCE (TEMP,TEMPB(1))
      COMMON/TX/TXBUF
      DO 20 J=1,2
      TEMP=TX(1,J)
      TXBUF(1,J)=TEMPB(1)
      DO 10 I=2,TXBUF(1,J),2
      TEMP=TX(I/2+1,J)
      TXBUF(I,J)=TEMPB(1)
10    TXBUF(I+1,J)=TEMPB(2)
20    CONTINUE
      RETURN
      END
C
```

```
C
C
C       DUMMY ROUTINES TO DIRECT I/O TO CORRECT LINK
C
C
C
      SUBROUTINE RXIHA2
      CALL LKANMY
      CALL RXIPRO(1)
      RETURN
      END
C
C
C
      SUBROUTINE RXIHA3
      CALL LKANMY
      CALL RXIPRO(2)
      RETURN
      END
C
C
C
      SUBROUTINE TXIHA2
      CALL LKANMY
      CALL TXIPRO(1)
      RETURN
      END
C
C
C
      SUBROUTINE TXIHA3
      CALL LKANMY
      CALL TXIPRO(2)
      RETURN
      END
```

```
        SUBROUTINE RK(IVAL)
        INTEGER*2 SIZE(2),RXB(128,2),SM2,LAST1(2),NEXT1(2),MBUFS(1000,1
        INTEGER*2 SECS
        BYTE ERROR(2)
        LOGICAL*1 TXFREE(2),RXFREE(2),MBFREE(2)
        COMMON/FLAGS/RXFREE,TXFREE,ERROR
        COMMON/MBUFS/MBUFS,MBFREE,LAST1,NEXT1,SIZE,SECS
C
C ASYNCHRONOUS COMPLETION ROUTINE TO MAINTAIN THE MAIN
C BUFFER(S) FOR DATA RECEIVED OVER THE I-MIC LINKS.
C
C
C
        DO 1000  I=1,2
C
C  ERROR FLAG DISABLES LINK
C
        IF(ERROR(I) .EQ. 0)  GOTO 150
        TYPE 3,I,ERROR(I)
        GOTO 1000
C
C  CHECK FOR NEW DATA
C
150     IF(RXFREE(I))  GOTO 1000
C
C HAVE NEW DATA ON LINK I.
C MAIN BUFFER ACCESS IN PROGRESS?
C
        IF(.NOT. MBFREE(I))  GOTO 1000
C
C NO. SO UPDATE MAIN BUFFER.
C
        CALL RXFORD(RXB)
        SM2=SIZE(I)-2
        NBYTES=RXB(1,I)
        NWORDS=NBYTES/2
        IF(NWORDS .LE. SM2)  GOTO 180
        TYPE 2,I,NWORDS,SM2
        GOTO 190
180     IF(NWORDS*2 .EQ. NBYTES)  GOTO 100
        TYPE 1,I,NBYTES
        RXB(1,I)=2*NWORDS
C
C NEXT1(I) POINTS TO THE NEXT SPARE WORD IN MBUF1.
C LW1 WILL BE THE LAST WORD WRITTEN IN THIS UPDATE.
C
        TYPE 200,(RXB(L,I),L=1,NWORDS)
200     FORMAT(1H ,10I6)
100     LW1=NEXT1(I)+NWORDS+1

C
C OFF END OF MAIN BUFFER?
C
        IF(LW1 .LE. SIZE(I))  GOTO 110
```

```
C
C YES. SO MUST SHIFT OLDEST DATA BLOCK OFF LEFT-HAND END
C TO MAKE ROOM.
C
        K1=MBUFS(1,I)
        KR=K1-1
        L=0
C
C K1 POINTS TO THE FIRST WORD OF THE NEXT DATA BLOCK
C KR = THE NUMBER OF WORDS RELEASED BY THIS SHIFT.
C FINISHED THIS SHIFT?
C
170     IF(K1 .EQ. NEXT1(I))  GOTO 160
C NOT YET.
        K2=MBUFS(K1,I)-1
C K2 POINTS TO THE LAST WORD OF THE NEXT DATA BLOCK.
C SHIFT IT DOWN.
        L1=L+1
        DO 120  K=K1,K2
        L=L+1
        MBUFS(L,I)=MBUFS(K,I)
120     CONTINUE
C UPDATE CHAIN LINKS.
        MBUFS(L1,I)=MBUFS(L1,I)-KR
        L1=L1+1
        MBUFS(L1,I)=MBUFS(L1,I)-KR
C UPDATE K1 AND SEE IF WE HAVE SHIFTED ALL CURRENT DATA BLOCKS.
        K1=K2+1
        GOTO 170
C
C FINISHED THIS SHIFT, RELEASING KR WORDS.
C UPDATE NEXT1(I) AND LAST1(I) AND SEE IF WE NOW HAVE ROOM FOR
C THE NEW DATA.
C
160     NEXT1(I)=NEXT1(I)-KR
        LAST1(I)=LAST1(I)-KR
        GOTO 100
C
C NOW HAVE ROOM FOR NEW DATA.
C ENTER NEW CHAIN LINKS.
C
110     K1=NEXT1(I)
        MBUFS(K1,I)=LW1+1
        K1=K1+1
        MBUFS(K1,I)=LAST1(I)+1
        K1=K1+1
C
C ENTER NEW DATA.
C
        L=0
        DO 130  K=K1,LW1
        L=L+1
        MBUFS(K,I)=RXB(L+1,I)
130     CONTINUE
```

```
C
C UPDATE POINTERS AND RELEASE RXBUF FOR NEW DATA.
C
        LAST1(I)=NEXT1(I)
        NEXT1(I)=LW1+1
190     RXFREE(I)=.TRUE.
C
1000    CONTINUE
C
C
C
C ALL LINKS DONE.
C TIDY UP AND RETURN
C
        CALL IBLFFX

        RETURN
C
1       FORMAT('OS/R CRBUFS - I-MIC LINK #',I1/
     +  ' NBYTES ODD-VALUED AT ',I3/
     +  ' LAST DATA BYTE WILL BE DISCARDED.'//)
2       FORMAT('OS/R CRBUFS - I-MIC LINK #',I1/
     +  1X,I6,' WORDS RECEIVED. BUFFER ALLOWS ONLY ',I6/
     +  ' DATA THEREFORE DISCARDED.'//)
3       FORMAT('OS/R CRBUFS - I-MIC LINK #',I1/
     +  ' ERROR FLAG SET TO ',I2/
     +  ' LINK IS DISABLED.'//)
        END
```

B2.1 Coordinator Software Listing

Background Routines for IBMLF

```
R LINK
DK:IBMLFB=DK:IBMLFB,GET2,DATEO,TIME/C
IBMF23,LINKXD,RK,LKSET2/C
TSXLIB,SDIC,ICSUB,GRAFLB,TIDY//
```

```
      PROGRAM IBMLFB
C
C     INTERACTION BALANCE METHOD - LOCAL FEEDBACK
C
C     CO-ORDINATOR OPTIMISATION USING SIMPLEX ALGORITHM
C
C
C
      INTEGER*2 BUF(17),RECBF(3),REPLY,STOPFG
      INTEGER*2 IBGBF(2),TEMBUF(17),N,IC,ICOUNT,L,H,N1
      REAL*4 A(5),P(4,4),BGBF(2),STEP,ALPHA,BETA,GAMMA,FUN
      REAL*4 SS,T,VL,VH,SIZE,D,D1,DX(17),TOL,X,B(4)
      LOGICAL*1 START,CR
      BYTE DATAX,DATAR
      COMMON/TRANSF/BUF,IBGBF,START,DATAX,DATAR
      COMMON DX,RU11,RY11,RU21,RY21,RY22
      COMMON/ICFG/ICFG,STOPFG
C
C
C     DISABLE CONTROL 'C' TO AVOID CRASHING THE SYSTEM
C
      CALL SCCA(0)
C
      ICFG=0
      GOTO 1007
      CALL PRINT(' INTECOLOR DISPLAY? KEY IN: 1(YES) ; 0(NO)')
      ACCEPT 1009,ICFG
1009  FORMAT(I2)
C
1007  CALL ASSIGN(3,'IBMLF1.DAT',0,NEW,NC,1)
      CALL ASSIGN(4,'GRAPH.DAT',0,NEW,NC,1)
      CALL ASSIGN(6,'IBMLFB.DAT',0,NEW,NC,1)
      CALL ASSIGN(9,'CUGRA.DAT',0,NEW,NC,1)
C
C
C     INITIALISATION OF I-MIC LINKS
C
      CALL DATE0
      CALL TIME
      START=.TRUE.
      CALL LNKSET
      CALL TRMASC(30,85)
      CALL IPOKE('44,IPEEK('44).OR.'100)
C
      A(1)=0.0
      N=2
      STEP=0.1
      IC=N+2
      ALPHA=1.3
      BETA=0.5
      GAMMA=2.0
      TOL=0.0002
      ICOUNT=1
      DATAR=0
      DATAX=0
      STOPFG=0
      WRITE(6,8)
8     FORMAT('ICOUNT L1     L2     P.MIN    P1C     P2C     P1H     P2H'
      WRITE(3,804)
804   FORMAT(' ICOUNT    PMAX       PMID       PMIN  (PMAX-PMIN)  SSI
C
C     FIRST ESTIMATE OF SIMPLEX VERTICE AND ITS FUNCTION VALUE
```

```
      CALL GET1(TEMBUF,FUN,BGBF)
      IF (STOPFG.EQ.1) GOTO 140
      IF (FUN.LT.32.76) GOTO 15
      CALL PRINT(' ENTER ESTIMATE PRICE VARABLES, LAMDAS:L1,L2:- ')
      ACCEPT 16,BGBF(1),BGBF(2)
16    FORMAT(2F8.4)
      DO 17 L=1,N
      D=BGBF(L)*1000.0
      IBGBF(L)=INT(D)
17    CONTINUE
      CALL SYNCH
      CALL GET1(TEMBUF,FUN,BGBF)
      IF (STOPFG.EQ.1) GOTO 140
15    P(N+1,1)=FUN
18    DO 10 I=1,N
      P(I,1)=BGBF(I)
10    CONTINUE
      WRITE(6,99) ICOUNT,BGBF(1),BGBF(2),FUN
99    FORMAT(I3,2F7.3,F8.4,32X,':GEN')
      COUNT=FLOAT(ICOUNT)
      DY1=0.05
      DY2=0.2
      WRITE(4,910) COUNT,BGBF(1),BGBF(2),FUN,DY1,DY2
      WRITE(9,95)COUNT,DX(5),DX(6),DX(11),DX(12),DX(13),RY11,RY21,R
95    FORMAT(F5.1,8F7.3)
      L=1
C
C     GENERATE REST OF SIMPLEX VERTICES AND THEIR FUNCTION VALUES
C
      SIZE=0.0
      DO 20 J=2,N+1
      DO 25 I=1,N
      P(I,J)=BGBF(I)
25    CONTINUE
      T=STEP
      IF (J.NE.2) GOTO 600
      D=3.0-T
      IF (BGBF(1).LT.D) GOTO 27
601   P(J-1,J)=BGBF(J-1)-T
      GOTO 26
600   D=1.846-T
      IF (BGBF(2).GT.D) GOTO 601
27    P(J-1,J)=BGBF(J-1)+T
26    IF (P(J-1,J).NE.BGBF(J-1)) GOTO 21
      T=T*10.0
      GOTO 27
21    SIZE=SIZE+ABS(T)
20    CONTINUE
23    SS=SIZE
      DO 30 J=1,N+1
      IF (J.EQ.L) GOTO 30
      DO 35 I=1,N
      BGBF(I)=P(I,J)
35    CONTINUE
      DO 200 I=1,N
      D=BGBF(I)*1000.0
      IBGBF(I)=INT(D)
200   CONTINUE
      CALL SYNCH
      CALL GET1(TEMBUF,FUN,BGBF)
      IF (STOPFG.EQ.1) GOTO 140
      ICOUNT=ICOUNT+1
      COUNT=COUNT+1.0
      P(N+1,J)=FUN
      IF (ICOUNT .GT. 3) GOTO 198
```

```
      WRITE(6,99) ICOUNT,BGBF(1),BGBF(2),FUN
      WRITE(4,910) COUNT,BGBF(1),BGBF(2),FUN,DY1,DY2
      WRITE(9,95)COUNT,DX(5),DX(6),DX(11),DX(12),DX(13),RY11,RY21,RY22
      GOTO 30
198   WRITE(6,2)ICOUNT,BGBF(1),BGBF(2),FUN,P(1,4),P(2,4),P(1,H),P(2,H)
2     FORMAT(I3,2F7.3,F8.4,4F7.3,4X,':GEN')
      WRITE(3,1234) ICOUNT,VH,P(N+1,N1),VL,X,SS
      WRITE(4,910) COUNT,BGBF(1),BGBF(2),FUN,X,SS
910   FORMAT(F5.1,2F7.3,F8.4,2F10.5)
      WRITE(9,95)COUNT,DX(5),DX(6),DX(11),DX(12),DX(13),RY11,RY21,RY22
30    CONTINUE
C
C     PERFORM TESTS AND RE-ORDER FUNCTION VALUES
C
33    CONTINUE
      DO 40 I=1,N+1
40    B(I)=P(N+1,I)
      DO 41 I=1,N
      IJ=I+1
      DO 42 J=IJ,N+1
      IF (B(I).LE.B(J)) GOTO 42
      T=B(I)
      B(I)=B(J)
      B(J)=T
42    CONTINUE
41    CONTINUE
      DO 43 I=1,N+1
      DO 44 J=1,N+1
      IF (P(N+1,J).NE.B(I)) GOTO 44
      IF (I.EQ.1) GOTO 45
      IF (I.EQ.N) GOTO 46
      H=J
      GOTO 43
45    L=J
      GOTO 43
46    N1=J
44    CONTINUE
43    CONTINUE
      VL=P(N+1,L)
      VH=P(N+1,H)
      X=VH-VL
      TYPE 1234,ICOUNT,VH,P(N+1,N1),VL,X,SS
      WRITE(3,1234) ICOUNT,VH,P(N+1,N1),VL,X,SS
1234  FORMAT(I5,5F10.5)
      WRITE(4,910) COUNT,BGBF(1),BGBF(2),FUN,X,SS
      IF (X.LE.TOL) GOTO 150
      IF (ICOUNT .LT. 30) GOTO 3456
      CALL TIME
      TYPE 3457
3457  FORMAT(' TYPE 0 (STOP) OR 1 (CONTINUE)')
      ACCEPT 3460,REPLY
3460  FORMAT(I3)
      IF (REPLY .EQ. 1) GOTO 3456
      GOTO 150
3456  D1=0.6*VH
      IF (VL.GT.D1) GOTO 49
      P(3,1)=P(3,L)
      BGBF(1)=P(1,L)
      BGBF(2)=P(2,L)
      GOTO 18
49    DO 50 I=1,N
      T=-P(I,H)
      DO 55 J=1,(N+1)
      T=T+P(I,J)
55    CONTINUE
      P(I,IC)=T/FLOAT(N)
```

```
50      CONTINUE
C
C       REFLECTION
C
        DO 60 I=1,N
        BGBF(I)=(1.0+ALPHA)*P(I,IC)-ALPHA*P(I,H)
60      CONTINUE
        DO 201 I=1,N
        D=BGBF(I)*1000.0
        IBGBF(I)=INT(D)
201     CONTINUE
        CALL SYNCH
        CALL GET1(TEMBUF,FUN,BGBF)
        IF (STOPFG.EQ.1) GOTO 140
        ICOUNT=ICOUNT+1
        COUNT=COUNT+1.0
        WRITE(3,1234) ICOUNT,VH,P(N+1,N1),VL,X,SS
        WRITE(4,910) COUNT,BGBF(1),BGBF(2),FUN,X,SS
        WRITE(6,3) ICOUNT,BGBF(1),BGBF(2),FUN,P(1,4),P(2,4),P(1,H),P(2,H)
3       FORMAT(I3,2F7.3,F8.4,4F7.3,4X,':REF')
        WRITE(9,95)COUNT,DX(5),DX(6),DX(11),DX(12),DX(13),RY11,RY21,RY22
        IF (FUN.LT.VL) GOTO 100
        IF (FUN.LT.P(N+1,N1)) GOTO 130
        IF (FUN.GE.VH) GOTO 80
        DO 70 I=1,N
        P(I,H)=BGBF(I)
70      CONTINUE
        P(N+1,H)=FUN
C
C       REDUCTION
C
80      DO 82 I=1,N
        BGBF(I)=(1.0-BETA)*P(I,H)+BETA*P(I,IC)
82      CONTINUE
        DO 202 I=1,N
        D=BGBF(I)*1000.0
        IBGBF(I)=INT(D)
202     CONTINUE
        CALL SYNCH
        CALL GET1(TEMBUF,FUN,BGBF)
        IF (STOPFG.EQ.1) GOTO 140
        ICOUNT=ICOUNT+1.0
        COUNT=COUNT+1.0
        WRITE(3,1234) ICOUNT,VH,P(N+1,N1),VL,X,SS
        WRITE(4,910) COUNT,BGBF(1),BGBF(2),FUN,X,SS
        WRITE(6,5) ICOUNT,BGBF(1),BGBF(2),FUN,P(1,4),P(2,4),P(1,H),P(2,H)
5       FORMAT(I3,2F7.3,F8.4,4F7.3,4X,':RED')
        WRITE(9,95)COUNT,DX(5),DX(6),DX(11),DX(12),DX(13),RY11,RY21,RY22
        IF (FUN.LT.P(N+1,H)) GOTO 130
C
C       CONTRACTION
C
        SIZE=0.0
        DO 90 J=1,N+1
        IF (J.EQ.L) GOTO 90
        DO 92 I=1,N
        P(I,J)=BETA*(P(I,J)-P(I,L))+P(I,L)
        SIZE=SIZE+ABS(P(I,J)-P(I,L))
92      CONTINUE
90      CONTINUE
        IF (SIZE.GE.0.0001) GOTO 446
        GOTO 150
446     IF (SIZE.LT.SS) GOTO 23
        CALL PRINT(' SIMPLEX FAILS TO CONTRACT')
        WRITE(6,444) SIZE,SS
444     FORMAT(//,2F12.5,///,'          SIMPLEX FAILS TO CONTRACT!',///)
```

```
      GOTO 23
C
C     EXTENSION
C
100   DO 102 I=1,N
      T=GAMMA*BGBF(I)+(1.0-GAMMA)*F(I,IC)
      F(I,IC)=BGBF(I)
      BGBF(I)=T
102   CONTINUE
      F(N+1,IC)=FUN
      DO 203 I=1,N
      D=BGBF(I)*1000.0
      IBGBF(I)=INT(D)
203   CONTINUE
      CALL SYNCH
      CALL GET1(TEMBUF,FUN,BGBF)
      IF (STOPFG.EQ.1) GOTO 140
      ICOUNT=ICOUNT+1
      COUNT=COUNT+1.0
      WRITE(3,1234) ICOUNT,VH,F(N+1,N1),VL,X,SS
      WRITE(4,910) COUNT,BGBF(1),BGBF(2),FUN,X,SS
      WRITE(6,4) ICOUNT,BGBF(1),BGBF(2),FUN,F(1,4),F(2,4),F(1,H),F(2,H)
4     FORMAT(I3,2F7.3,F8.4,4F7.3,4X,':EXT')
      WRITE(9,95)COUNT,DX(5),DX(6),DX(11),DX(12),DX(13),RY11,RY21,RY22
      IF (FUN.LT.F(N+1,IC)) GOTO 130
      DO 110 I=1,N
      BGBF(I)=F(I,IC)
110   CONTINUE
      FUN=F(N+1,IC)
130   DO 112 I=1,N
      F(I,H)=BGBF(I)
112   CONTINUE
      F(N+1,H)=FUN
      GOTO 33
150   DO 151 J=1,N
      BGBF(J)=F(J,L)
      D=BGBF(J)*1000.0
151   IBGBF(J)=INT(D)
      CALL SYNCH
      CALL GET1(TEMBUF,FUN,BGBF)
      IF (STOPFG.EQ.1) GOTO 140
      ICOUNT=ICOUNT+1
      COUNT=COUNT+1.0
      WRITE(3,1234) ICOUNT,VH,F(N+1,N1),VL,X,SS
      WRITE(4,910) COUNT,BGBF(1),BGBF(2),FUN,X,SS
      WRITE(6,6) ICOUNT,BGBF(1),BGBF(2),FUN,F(1,4),F(2,4),F(1,H),F(2,H)
6     FORMAT(I3,2F7.3,F8.4,4F7.3)
      WRITE(9,95)COUNT,DX(5),DX(6),DX(11),DX(12),DX(13),RY11,RY21,RY22
C
C     SOLUTION OUTPUT
C
143   WRITE(6,154) ICOUNT
154   FORMAT(//,' AFTER',I4,2X,'FUNCTION CALLS',/)
      WRITE(6,153) BGBF(1),BGBF(2)
153   FORMAT(' LAMDA1=',F7.3,7X,'LAMDA2=',F7.3,/)
      WRITE(6,501) DX(5),DX(6)
501   FORMAT(' C11= ',F7.3,5X,'C12= ',F7.3)
      WRITE(6,502) RU11,RY11
502   FORMAT(' RU11=',F7.3,5X,'RY11=',F7.3,/)
      WRITE(6,503) DX(11),DX(12),DX(13)
503   FORMAT(' C21= ',F7.3,5X,'C22= ',F7.3,5X,'C23= ',F7.3)
      WRITE(6,504) RU21,RY21,RY22
504   FORMAT(' RU21=',F7.3,5X,'RY21=',F7.3,5X,'RY22=',F7.3,/)
      WRITE(6,505) FUN
505   FORMAT(' FINAL REAL FUNCTIONAL VALUE IS   ',F8.4,//)
      IBGBF(1)=32766
```

```
        IBGBF(2)=32766
        CALL SYNCH
140     CALL TIME
        TYPE 678
678     FORMAT(1H ,' PLEASE STOP I-MICS AND THEN TYPE <CR>')
        ACCEPT 679,CR
679     FORMAT(A1)
        CALL TIDY2
        STOP
        END
C
C
C
        SUBROUTINE SYNCH
C
C       THIS SUBROUTINE SYNCHRONISES THE DATA TRANSFER BETWEEN
C       THE F/G & B/G PROGRAM
C
C
        INTEGER*2 BUF(17),IBGBF(2)
        LOGICAL*1 START
        BYTE DATAX,DATAR
        COMMON/TRANSF/BUF,IBGBF,START,DATAX,DATAR
C
C
        DATAR=1
20      IF(DATAR.EQ.2)GOTO 10
        GOTO 20
10      DATAR=0
        RETURN
        END
```

```
C       PROGRAM GET2
C
        SUBROUTINE GET1(TEMB,FUN1,GBF)
C
C       THIS SUBROUTINE GETS DATA FROM F/G PROGRAM (DECISION UNITS) TO THE
C       B/G PROGRAM (COORDINATOR) AND OUTPUT THE DATA ON INTECOLOR DISPLAY
C
C
        INTEGER*2 TEMB(17),IBGBF(2),BUF(17),STOPFG,SHUT
        REAL*4 A1(3),DX(17),FUN1,GBF(2),DU(2),
     1         RU11,RU21,RY11,RY21,RY22,XY(20)
        LOGICAL*1 START
        BYTE DATAX,DATAR
        COMMON/TRANSF/BUF,IBGBF,START,DATAX,DATAR
        COMMON/ICF/ICFLAG,XY
        COMMON/ICFG/ICFG,STOPFG
        COMMON DX,RU11,RY11,RU21,RY21,RY22
        DATA YES/1HY/
C
C
        A1(1)=0.0
4       CONTINUE
C
C       CHECK IF INTECOLOR DISPLAY FLAG IS SET
C
        IF (ICFG.EQ.0) GOTO 200
C
C       OUTPUT DATA TO INTECOLOR
C
        IF (ICFLAG.EQ.0) GOTO 200
        CALL SDIC(XY)
        ICFLAG=0
C
C       CHECK IF WANT TO SHUT DOWN THE SYSTEM
C
200     IVAL=ITTINR(0)
        IF (IVAL.LT.0) GOTO 210
        READ(5,60) AVAL
60      FORMAT(A1)
        IF (IVAL.NE.83) GOTO 210
        CALL CLEAR
        TYPE 70
70      FORMAT(' SYSTEM SHUT DOWN?',/)
        TYPE 80
80      FORMAT(' 0 : NO',/,' 1 : IMMEDIATELY',/,' 2 : AFTER NEXT DATA')
        ACCEPT 90,SHUT
90      FORMAT(/,' TYPE IN : ',I2)
        IF (SHUT.EQ.0) GOTO 210
        STOPFG=1
        IF (SHUT.EQ.2) GOTO 210
        CALL TIDY2
        RETURN
210     IF(DATAX.EQ.0)GOTO 4
C
C       SEND DATA TO B/G PROGRAM
C
        DATAX=2
        DO 5 J=1,16
5       TEMB(J+1)=BUF(J)
        DO 3 K=1,16
        DX(K)=FLOAT(TEMB(K))
        DX(K)=DX(K)/1000.0
```

```
3       CONTINUE
        RU11=DX(8)
        RU21=DX(16)
        RY11=RU21
        RY21=RU11
        RY22=2.3*DX(12)-0.7*DX(13)-1.1*RU21
        DU(1)=(RY11-1.0)**2+DX(5)**2+DX(6)**2
        DU(2)=2.0*(RY21-2.0)**2+(RY22-3.0)**2+DX(11)**2+DX(12)**2
        DU(2)=DU(2)+DX(13)**2
        FUN1=DU(1)+DU(2)
        GBF(1)=DX(3)
        GBF(2)=DX(9)
        RETURN
        END
```

```
      SUBROUTINE IBMF23
C
C     THIS SUBROUTINE IS THE TIME CRITICAL F/G PROGRAM WHICH USED
C     TO COMMUNICATE SYNCHRONISELY WITH THE I-MIC'S THROUGH THE LINKS.
C
C
      INTEGER*2 BUF(17),BUF1(17,3),RECBF(3),NEXT1(2),ICOUNT(2)
      INTEGER*2 IBGBF(2),UDF(2)
      INTEGER*2 LAST1(2),MBUFS(1000,2),YES,REPLY,TXB(128,2),DATDU(2)
      REAL*4 XY(17)
      BYTE DATAX,DATAR,TFLAG
      LOGICAL*1 TXFREE(2),MBFREE(2),START,RXFREE(2)
      COMMON/FLAGS/RXFREE,TXFREE
      COMMON/MBUFS/MBUFS,MBFREE,LAST1,NEXT1
      COMMON/TRANSF/BUF,IBGBF,START,DATAX,DATAR
      COMMON/ICF/ICFLAG,XY
      COMMON/ICFG/ICFG
      DATA YES/1HY/
C
C
C     INITIALISATION
C
      IF(.NOT.START)GOTO 235
C
C...I --- THE LOCAL DECISION UNIT NUMBER
C
      ICFLAG=0
      ICC=1
      IDU=1
      DATDU(1)=6
      DATDU(2)=8
200   DO 111 K=1,2
      UDF(K)=0
111   ICOUNT(K)=0
      K0=0
      I=1
      TFLAG=0
      START=.FALSE.
235   IF(TFLAG.EQ.1)GOTO 2080
      IF(TFLAG.EQ.2)GOTO 2090
      UDF(1)=0
      UDF(2)=0
      DO 201 K=1,16
201   BUF1(K,I)=0
C
C...TEST IF ANY DATA RECEIVED FROM IMIC
      IF(LAST1(I) .LT. 0)  GOTO 995
      JC=LAST1(I)+2
C
C...TEST IF NEW BLOCK OF DATA RECEIVED
C    AS A NEW BLOCK NUMBER IS TRANSFERRED
C    FROM THE IMIC WHEN TRANSFER TAKES PLACE
      IF(MBUFS(JC,I).GT.ICOUNT(I))GO TO 240
      GOTO 995
C
C...MBFREE(I) FLAG IS TO CONTROL THE UPDATING
C    OF THE MASTER BUFFER
240   MBFREE(I)=.FALSE.
C
C...SELECTS LATEST DATA BLOCK FROM MASTER BUFFER
      K1=1
110   K2=MBUFS(K1,I)-1
```

```
          K1=K2+1
          IF(K1 .LE. LAST1(I))  GOTO 110
150       J=LAST1(I)
          K0=K0+1
          BUF1(1,I)=K0
          JC=J+2
          JS=J+4
          JF=JS+DATIU(I)
          KJ=1
C
C...SETS LATEST DATA VALUES INTO TEMPORARY BUFFER
C    WHICH IS USED TO SEND DATA TO THE BACKGROUND
C    PROGRAM
          DO 50 K=JS,JF
          KJ=KJ+1
          BUF1(KJ,I)=MBUFS(K,I)
50        CONTINUE
C
          IF (ICFG.EQ.1) GOTO 156
          IF (I.EQ.2) GOTO 2001
          TYPE 2000,I,(BUF1(K,I),K=2,6)
2000      FORMAT(I2,5I6)
          GOTO 156
2001      TYPE 2002,I,(BUF1(K,I),K=2,8)
2002      FORMAT(I2,7I6)
C
156       DO 157 K=1,2
157       BUF1(K,I)=MBUFS(JC+K,I)
          MBFREE(I)=.TRUE.
          ICOUNT(I)=MBUFS(JC,I)
          IF (I .EQ. 2) GOTO 2020
          I=2
          GOTO 235
C
C         STORE DATA IN A TEMPORARY ARRAY WHICH WILL BE OUTPUT
C         ON THE INTECOLOR DISPLAY
C
2020      DO 10 K=1,16
10.       XY(K)=0
          XY(1)=FLOAT(ICC)
          XY(2)=FLOAT(IDU)
          DO 20 K=3,8
20        XY(K)=FLOAT(BUF1(K-1,1))/1000.0
          DO 30 K=9,16
30        XY(K)=FLOAT(BUF1(K-7,2))/1000.0
          IDU=IDU+1
C
C         CHECK IF THE DATA RECEIVED FROM THE DECISION UNITS
C         ARE CONVERGED SOLUTION
C
          IF(BUF1(1,1).EQ.1 .AND. BUF1(1,2).EQ.1) GOTO 2030
          IF(BUF1(1,1).EQ.0) GOTO 479
          UDF(2)=1
479       IF(BUF1(1,2).EQ.0) GOTO 480
          UDF(1)=1
C
C         NOT CONVERGED SOLUTION, SEND THE DATA BACK TO
C         CORRESPONDING DECISION UNITS
C
480       NWORD=5
          BUF1(4,2)=BUF1(3,1)
          BUF1(3,1)=BUF1(2,2)
          BUF1(4,1)=BUF1(3,2)
          BUF1(3,2)=BUF1(2,2)
          BUF1(2,2)=BUF1(2,1)
          BUF1(5,1)=UDF(1)
```

```
        BUF1(5,2)=UDF(2)
        DO 2010 I=1,2
        DO 2015 K=1,NWORD
2015    TXB(K+1,I)=BUF1(K+1,I)
270     IF (TXFREE(I)) GOTO 225
        CALL PRINT(' TXBUF NOT FREE')
        GOTO 995
225     TXB(1,I)=2*NWORD
        CALL TXFORD(TXB)
        TXFREE(I)=.FALSE.
2010    CONTINUE
        ICFLAG=1
        I=1
        UDF(1)=0
        UDF(2)=0
        GOTO 995
C
C
C       STORE THE DATA OF THE CONVERGED DECISION UNIT SOLUTIONS INTO A
C       TEMPORARY BUFFER READY TO TRANSFER TO THE B/G PROGRAM
C
2030    IDU=1
        UDF(1)=1
        UDF(2)=1
        DO 1500 K=1,16
1500    BUF(K)=0
        BUF(1)=ICC
        DO 1501 K=2,7
1501    BUF(K)=BUF1(K,1)
        DO 1502 K=8,15
1502    BUF(K)=BUF1(K-6,2)
        IF (ICFG.EQ.1) GOTO 9997
        TYPE 9999,(BUF(K),K=1,15)
9999    FORMAT(15I6)
C
C       SET ALL THE NECESSARILY FLAGS TO SYNCHRONISE THE F/G & B/G PROGRAI
C
9997    TFLAG=1
2080    IF(DATAX.EQ.2)GOTO 2070
        DATAX=1
        GOTO 995
2070    DATAX=0
C
C...SETS-UP TEMPORARY BUFFER TO RECEIVE
C   PARAMETER DATA FROM THE BACKGROUND PROGRAM
        DO 100 KM=1,3
        RECBF(KM)=0
100     CONTINUE
        TFLAG=2
2090    IF(DATAR.EQ.0)GOTO 995
        DATAR=2
        DO 2060 IJK=1,2
2060    RECBF(IJK)=IBGBF(IJK)
C
C...TXFREE(I) FLAG IS USED TO CONTROL
C   THE TRANSMISSION OF DATA TO THE IMIC
C   AS DATA MAY ONLY BE TRANSFERRED AFTER
C   THE IMIC HAS COMPLETED A TRANSFER
        DO 9998 I=1,2
170     IF(TXFREE(I))  GOTO 125
        CALL PRINT(' TXBUF NOT FREE ')
        GOTO 235
125     CONTINUE
C
C...NWORDS....NUMBER OF PARAMETER DATA VALUES
        NWORDS=5
```

```
            DO 126 K=1,2
126         TXB(K+1,I)=RECBF(K)
            DO 127 K=4,6
127         TXB(K,I)=UDF(I)
            TXB(1,I)=2*NWORDS
            CALL TXFORD(TXB)
            TXFREE(I)=.FALSE.
9998        CONTINUE
C
            ICC=ICC+1
            UDF(1)=0
            UDF(2)=0
            I=1
            TFLAG=0
995         RETURN
140         END
```

```
      SUBROUTINE SDIC(DX)
C
C     THIS SUBROUTINE OUTPUT DATA FOR INTECOLOR DISPLAY
C
C
      INTEGER*2 IOP(17)
      REAL*4 DX(17),OP(17),TY11,TY21
C
C
      TY21=(0.66*DX(6)-1.54*DX(5)-1.3*DX(11)+1.1*DX(12))/0.98
      TY11=1.4*DX(5)-0.6*DX(6)+1.8*TY21
      IOP(1)=INT(DX(2))
      OP(2)=DX(3)
      OP(3)=DX(4)
      OP(4)=DX(5)
      OP(5)=DX(6)
      OP(6)=TY11
      OP(7)=(TY11-1.0)**2+DX(5)**2+DX(6)**2
      OP(8)=DX(9)
      OP(9)=DX(10)
      OP(10)=DX(11)
      OP(11)=DX(12)
      OP(12)=DX(13)
      OP(13)=TY21
      OP(14)=2.3*DX(12)-0.7*DX(13)-1.1*TY11
      OP(15)=2.0*(TY21-2.0)**2+(OP(14)-3.0)**2
      OP(15)=OP(15)+DX(11)**2+DX(12)**2+DX(13)**2
      OP(16)=OP(7)+OP(15)
      IOP(17)=INT(DX(1))
C
      DO 10 J=2,16
10    IOP(J)=INT(OP(J)*1000.0)
      CALL ICSUB(IOP,17)
      RETURN
      END
```

```
      SUBROUTINE ICSUB(OP,NN)
C
C     THIS SUBROUTINE SENDS DATA FROM B/G PROGRAM TO THE INTECOLOR
C
C
      REAL*8 SEND,B(17)
      INTEGER*2 OP(17)
      BYTE BUFFX(17)
      EQUIVALENCE (BUFFX(1),SEND)
C
C
      N=8
      DO 50 I=1,NN
      ENCODE(5,99,B(I)) OP(I)
50    CONTINUE
99    FORMAT(17I8)
      DO 200 I=1,NN
      SEND=B(I)
      CALL ARRAYX(N,BUFFX)
200   CONTINUE
      RETURN
      END
```

```
C
C       SUBROUTINE TIDY2.FOR ETC.
C
        SUBROUTINE TRMASC(I,J)
        BYTE I,J
        WRITE(7,1)I,J
1       FORMAT(1H+,72A1)
        RETURN
        END
C
C
        SUBROUTINE CLEAR
        COMMON/CLEAR/ICLEAR
        IF(ICLEAR.EQ.1) GOTO 10
C
C       ICLEAR=1 FOR ADM5
C
        IF(ITSLIN().NE.0) GOTO 20
        CALL TRMASC(27,12)
        CALL ISLEEP(0,0,0,10)
        RETURN
10      CALL TRMASC(26,0)
        RETURN
20      CALL TKERAS
        RETURN
        END
C
C       TIDY2 DISABLE THE RX AND TX INTERRUPTS SET BY LINK OPERATIN
C       ROUTINES
C
        SUBROUTINE TIDY2
        INTEGER*2 IVEC(2),RCSR(2),RBUF(2),XBUF(2),XCSR(2)
        COMMON/LCHAR/IVEC,RCSR,RBUF,XBUF,XCSR
        CALL MPIOPS
        CALL IPOKE(XCSR(1),0)
        CALL IPOKE(XCSR(2),0)
        CALL IPOKE(RCSR(1),0)
        CALL IPOKE(RCSR(2),0)
        RETURN
        END
C
        SUBROUTINE TKERAS
        INTEGER*2 TEMP,XBUF,XCSR
        BYTE TEMPB(2)
        EQUIVALENCE (TEMP,TEMPB(1))
        CALL MPIOPS
        ILIN=ITSLIN()
        IFLAG=0
        IF(ILIN.EQ.4) GOTO 10
        XCSR="176530+"10*ILIN+4
        GOTO 20
10      XCSR="177564
20      XBUF=XCSR+2
        TEMP=IPEEK(XCSR)
        IF(TEMPB(1).LT.128) GOTO 20
        IF(IFLAG.EQ.1) GOTO 30
        CALL IPOKE(XBUF,27)
        IFLAG=1
        GOTO 20
30      CALL IPOKE(XBUF,12)
        RETURN
        END
```

B2.2 Coordinator Software Listing

Background Routines for IBMGF

```
R LINK
DK:IBMGFB=DK:IBMGFB,IBGGET,DATEO,TIME/C
IBMGFF,LINKXD,RK,LKSET2/C
TSXLIB,ICIBGF,GRAFLB,TIDY//
```

```
      PROGRAM IBMGFB
C
C    INTERACTION BALANCE METHOD - GLOBAL FEEDBACK
C
C    CO-ORDINATOR OPTIMISATION USING STEEPEST DESCENT ALGORITHM
C
C
C
      INTEGER*2 N,IGRD,IC,ILAST,IT,CLF,IQ,RECBF(3),REPLY
      INTEGER*2 IBGBF(2),STOPFG,BUF(17)
      REAL*4 A(3),ITN(2),LA(2),TOL,B(4,4),FUN,C11,C12,U11,Y11
      REAL*4 C21,C22,C23,U21,Y21,Y22,X(2),C(2),S,T(2),Q,SLEN,D1
      REAL*4 D2,E(5),RU11,RY11,RU21,RY21,RY22,RFUN,H(4,4)
      LOGICAL*1 START,CR
      BYTE DATAX,DATAR
      COMMON/TRANSF/BUF,IBGBF,START,DATAX,DATAR
      COMMON/ICFG/ICFG,STOPFG
      COMMON/CONTOL/C11,C12,C21,C22,C23
      COMMON/INTER/U11,Y11,U21,Y21,Y22,RU11,RY11,RU21,RY21,RY22
      COMMON/OUT/FUN,RFUN,CLF,IC
C
C
C
C     DISABLE CONTROL 'C' TO AVOID CRASHING THE SYSTEM
C
      CALL SCCA(0)
C
      CALL PRINT(' INTECOLOR DISPLAY? KEY IN: 1(YES) ; 0(NO)')
      ACCEPT 1009,ICFG
1009  FORMAT(I2)
C
1091  CALL ASSIGN(6,'IBMGFB.DAT',0,NEW,NC,1)
      CALL ASSIGN(3,'GRAPH.DAT',0,NEW,NC,1)
      CALL ASSIGN(9,'CYGRA.DAT',0,NEW,NC,1)
C
C
C     INITIALISATION OF I-MIC LINKS
C
      WRITE(6,928)
928   FORMAT(' INTERACTION BALANCE METHOD WITH GLOBAL FEEDBACK (SY)',//
      CALL DATEO
      CALL TIME
      START=.TRUE.
      CALL LNKSET
      CALL TRMASC(30,85)
      CALL IPOKE('44,IPEEK('44).OR.'100)
C
      A(1)=0.0
      IQ=1
      CLF=1
      IC=1
      COUNT=FLOAT(IC)
      N=2
      IGRD=0
      SLEN=1.0
      TOL=0.002
      B(1,1)=2.0
      B(1,2)=0.5
      B(2,1)=0.6
      B(2,2)=1.5
      DATAR=0
```

```
      DATAX=0
      STOPFG=0
      WRITE(6,1005)
1005  FORMAT(' CAL    LA1      LA2     ITN1     ITN2     U11      Y11      U21
     *   Y21     Y22    RY11    RY21    RY22     MFI     RFI')
C
      CALL IBGGET(ITN,LA)
      IF (STOPFG.EQ.1) GOTO 20
C
C     STORE DATA IN OUTPUT FILES
C
      WRITE(6,1000) IC,LA(1),LA(2),ITN(1),ITN(2),U11,Y11,
     *                U21,Y21,Y22,RY11,RY21,RY22,FUN,RFUN
1000  FORMAT(I3,4F8.3,10F7.3)
      WRITE(3,1100) COUNT,LA(1),LA(2),ITN(1),ITN(2),FUN,RFUN
1100  FORMAT(F5.1,6F8.3)
      WRITE(9,1200) COUNT,C11,C12,C21,C22,C23,RY11,RY21,RY22
1200  FORMAT(F5.1,8F8.3)
C
      IC=IC+1
      COUNT=COUNT+1.0
      E(3)=SQRT(ITN(1)**2+ITN(2)**2)
C
      IGRD=IGRD+1
C
      IT=0
3     ILAST=IGRD
      H(IQ,1)=LA(1)
      H(IQ,2)=LA(2)
      H(IQ,3)=E(3)
      IQ=IQ+1
      IF (IQ.EQ.4) GOTO 500
C
4     DO 40 I=1,N
      X(I)=LA(I)
      C(I)=ITN(I)
40    CONTINUE
C
5     D1=0.0
      DO 50 I=1,N
      S=0.0
      DO 52 J=1,N
52    S=S+B(I,J)*ITN(J)
      T(I)=S
50    D1=D1+S*ITN(I)
C
      IF (D1.GT.0.0) GOTO 8
      IF (ILAST.EQ.IGRD) GOTO 18
      GOTO 3
C
8     DO 80 I=1,N
      LA(I)=X(I)+SLEN*T(I)
      IF (LA(I).NE.X(I)) GOTO 80
      IT=IT+1
80    CONTINUE
C     TYPE 88,X(1),X(2),C(1),C(2),T(1),T(2),LA(1),LA(2)
88    FORMAT(8F8.3)
C
      IF (IT.LT.N) GOTO 10
      IF (ILAST.EQ.IGRD) GOTO 18
      GOTO 3
500   Q=ABS(H(1,3)-H(3,3))
      IF (SLEN.LE.0.005) GOTO 600
      IF (Q.GT.TOL) GOTO 600
      LA(1)=(H(3,1)+H(2,1))/2.0
      LA(2)=(H(3,2)+H(2,2))/2.0
```

```
          IQ=3
          GOTO 10
600       DO 601 K3=1,2
          H(K3,1)=H(K3+1,1)
          H(K3,2)=H(K3+1,2)
          H(K3,3)=H(K3+1,3)
601     . CONTINUE
          IQ=3
          GOTO 4
C
10        DO 100 I=1,N
          E(I)=LA(I)*1000.0
          IBGBF(I)=INT(E(I))
100       CONTINUE
          CALL SYNCH
          CALL IBGGET(ITN,LA)
          IF (STOPFG.EQ.1) GOTO 20
          WRITE(6,1000) IC,LA(1),LA(2),ITN(1),ITN(2),U11,Y11,
      *                   U21,Y21,Y22,RY11,RY21,RY22,FUN,RFUN
          WRITE(3,1100) COUNT,LA(1),LA(2),ITN(1),ITN(2),FUN,RFUN
          WRITE(9,1200) COUNT,C11,C12,C21,C22,C23,RY11,RY21,RY22
          IC=IC+1
          COUNT=COUNT+1.0
C
11        E(3)=SQRT(ITN(1)**2+ITN(2)**2)
          IF (E(3).LE.TOL) GOTO 18
          SLEN=1.1*E(3)
          IF (SLEN.GT.0.005) GOTO 60
          SLEN=0.005
60        ITN(1)=ITN(1)/E(3)
          ITN(2)=ITN(2)/E(3)
C         TYPE 61,E(3),ITN(1),ITN(2),SLEN
61        FORMAT(4F10.3)
          GOTO 3
C
18        IF (CLF.EQ.2) GOTO 20
          CLF=2
          IQ=1
          IT=0
          SLEN=0.5
          CALL TIME
          TOL=0.0
          GOTO 3
C
C         SOLUTION OUTPUT
C
20        IC=IC-1
          WRITE(6,4500) IC
4500      FORMAT(//,' AFTER',I5,3X,'ITERATIONS',/)
          WRITE(6,4700) LA(1),LA(2)
4700      FORMAT(' LAMDA1=',F7.3,8X,'LAMDA2=',F7.3,/)
          WRITE(6,5000) C11,C12
5000      FORMAT(' C11=',F7.3,5X,'C12=',F7.3)
          WRITE(6,5001) U11,Y11
5001      FORMAT(' U11=',F7.3,5X,'Y11=',F7.3,/)
          WRITE(6,5200) C21,C22,C23
5200      FORMAT(' C21=',F7.3,5X,'C22=',F7.3,5X,'C23=',F7.3)
          WRITE(6,5201) U21,Y21,Y22
5201      FORMAT(' U21=',F7.3,5X,'Y21=',F7.3,5X,'Y22=',F7.3,/)
          WRITE(6,6000) FUN
6000      FORMAT(' FINAL FUNCTIONAL VALUE IS   ',F8.4,///)
          WRITE(6,6001) RU11,RY11
6001      FORMAT(' RU11=',F7.3,5X,'RY11=',F7.3,/)
          WRITE(6,6002) RU21,RY21,RY22
6002      FORMAT(' RU21=',F7.3,5X,'RY21=',F7.3,5X,'RY22=',F7.3,/)
          WRITE(6,6003) RFUN
```

```
6003    FORMAT(' REAL FINAL FUNCTIONAL VALUE IS    ',F8.4,//)
C
C       SEND TERMINATION SIGNAL TO COOORDINATOR
C
        IBGBF(1)=32766
        IBGBF(2)=32766
        CALL SYNCH
140     CALL TIME
        TYPE 678
678     FORMAT(1H ,' PLEASE STOP I-MICS AND THEN TYPE <CR>')
        ACCEPT 679,CR
679     FORMAT(A1)
        CALL TIDY2
        STOP
        END
C
C
C
        SUBROUTINE SYNCH
C
C       THIS SUBROUTINE SYNCHRONISES THE DATA TRANSFER BETWEEN
C       THE F/G & B/G PROGRAM
C
C
        INTEGER*2 BUF(17),IBGBF(2)
        LOGICAL*1 START
        BYTE DATAX,DATAR
        COMMON/TRANSF/BUF,IBGBF,START,DATAX,DATAR
C
C
        DATAR=1
20      IF(DATAR.EQ.2)GOTO 10
        GOTO 20
10      DATAR=0
        RETURN
        END
```

```
      SUBROUTINE IBGGET(ITN,LA)
C
CC    THIS SUBROUTINE GETS DATA FROM F/G PROGRAM (DECISION UNITS) TO THE
CC    B/G PROGRAM (COORDINATOR) AND OUTPUT THE DATA ON INTECOLOR DISPLAY
C
CC    USING INTERACTION BALANCE METHOD WITH GLOBAL FEEDBACK
CC
C
      INTEGER*2 BUF(17),IBGBF(2),STOPFG,CLF,SHUT,IDU(2),ERROR(2)
      INTEGER*2 BUF1(21),IRFUN
      REAL*4 A1(3),RFUN,FUN,LA(2),DU(2),RDU(2),
     1       RU11,RU21,RY11,RY21,RY22,BUFX(17),ITN(2)
      LOGICAL*1 START
      BYTE DATAX,DATAR
      COMMON/TRANSF/BUF,IBGBF,START,DATAX,DATAR
      COMMON/ICF/ICFLAG
      COMMON/ICFG/ICFG,STOPFG
      COMMON/CONTOL/C11,C12,C21,C22,C23
      COMMON/INTER/U11,Y11,U21,Y21,Y22,RU11,RY11,RU21,RY21,RY22
      COMMON/OUT/FUN,RFUN,CLF,IC
      DATA YES/1HY/
C
C
      A1(1)=0.0
4     CONTINUE
C
CC    CHECK IF INTECOLOR DISPLAY FLAG IS SET
C
      IF (ICFG.EQ.0) GOTO 200
C
CC    OUTPUT DATA TO INTECOLOR
C
      IF (ICFLAG.EQ.0) GOTO 200
C     TYPE 8,(BUF1(K),K=1,16),ERROR(1),ERROR(2),IDU(1),IDU(2),IRFUN
8     FORMAT(21I6)
      CALL ICIBGF(BUF1,ERROR,IDU,IRFUN)
      ICFLAG=0
C
CC    CHECK IF WANT TO SHUT DOWN THE SYSTEM
C
200   IVAL=ITTINR(0)
      IF (IVAL.LT.0) GOTO 210
      CALL CLEAR
      CALL PRINT(' ')
      CALL PRINT('           SYSTEM SHUT DOWN?')
      CALL PRINT(' TYPE IN:')
      CALL PRINT('           0 : NO')
      CALL PRINT('           1 : IMMEDIATELY')
      CALL PRINT('           2 : AFTER NEXT DATA SET')
      ACCEPT 90,SHUT
90    FORMAT(I2,/)
      IF (SHUT.EQ.0) GOTO 210
      STOPFG=1
      IF (SHUT.EQ.2) GOTO 210
      CALL TIDY2
      STOP
C
210   IF(DATAX.EQ.0)GOTO 4
C
CC    SEND DATA TO B/G PROGRAM
C
```

```
          DATAX=2
          DO 5 L=1,15
5         BUFX(L)=(FLOAT(BUF(L)))/1000.0
C
          LA(1)=BUFX(2)
          LA(2)=BUFX(8)
          C11=BUFX(4)
          C12=BUFX(5)
          U11=BUFX(3)
          Y11=BUFX(6)
          C21=BUFX(10)
          C22=BUFX(11)
          C23=BUFX(12)
          U21=BUFX(9)
          Y21=BUFX(13)
          Y22=BUFX(14)
          RU11=BUFX(7)
          RU21=BUFX(15)
          RY11=RU21
          RY21=RU11
          RY22=2.3*C22-0.7*C23-1.1*RU21
C
          IF (CLF.EQ.2) GOTO 999
C
C         OPEN LOOP INTERCONNECTION ERROR
C
          ITN(1)=U11-Y21
          ITN(2)=U21-Y11
          GOTO 1000
C
C         CLOSED LOOP INTERCONNECTION ERROR
C
999       ITN(1)=U11-U21-RY21+RY11
          ITN(2)=U21-2.0*U11-RY11+2.0*RY21
C
C         DETERMINATION OF MODEL PERFORMANCE
C
1000      DU(1)=(Y11-1.0)**2+C11**2+C12**2+LA(1)*U11-LA(2)*Y11
          DU(2)=2.0*(Y21-2.0)**2+(Y22-3.0)**2+C21**2+C22**2+C23**2
          DU(2)=DU(2)+LA(2)*U21-LA(1)*Y21
          FUN=DU(1)+DU(2)
C
C         DETERMINATION OF REAL PERFORMANCE
C
          RDU(1)=(RY11-1.0)**2+C11**2+C12**2+LA(1)*RU11-LA(2)*RY11
          RDU(2)=2.0*(RY21-2.0)**2+(RY22-3.0)**2+C21**2+C22**2
          RDU(2)=RDU(2)+C23**2+LA(2)*RU21-LA(1)*RY21
          RFUN=RDU(1)+RDU(2)
C
C         SET UP A BUFFER OF DATA TO BE DISPLAY ON INTECOLOR
C
          DO 1002 K=1,15
1002      BUF1(K)=BUF(K)
          BUF1(16)=INT(RY22*1000.0)
          DO 1001 K=1,2
          ERROR(K)=INT(ITN(K)*1000.0)
1001      IDU(K)=INT(DU(K)*1000.0)
          IRFUN=INT(RFUN*1000.0)
C
C         DISPLAY OF DATA FROM I-MICS
C
          TYPE 1,IC,ITN(1),ITN(2)
1         FORMAT(I3,' ITERATION    E1=',F7.3,'    E2=',F7.3)
          TYPE 2,(BUFX(K),K=2,7)
2         FORMAT(' DU1:  ',6F7.3)
          TYPE 3,(BUFX(K),K=8,15),RY22
```

```
3       FORMAT(' DU2:   ',9F7.3)
        TYPE 6,FUN,RFUN
6       FORMAT(' MRI=',F7.3,'    RPI='F7.3,/)
        RETURN
        END
```

```
      SUBROUTINE IBMGFF
C
C     THIS SUBROUTINE IS THE TIME CRITICAL F/G PROGRAM WHICH USED
C     TO COMMUNICATE SYNCHRONISELY WITH THE I-MIC'S THROUGH THE LINKS.
C
C     USING INTERACTION BALANCE METHOD WITH GLOBAL FEEDBACK
C
C
      INTEGER*2 BUF(17),IBGBF(2),RECBF(3),NEXT1(2),ICOUNT(2)
      INTEGER*2 LAST1(2),MBUFS(1000,2),YES,REPLY,TXB(128,2),DATDU(2)
      BYTE DATAX,DATAR,TFLAG
      LOGICAL*1 TXFREE(2),MBFREE(2),START,RXFREE(2)
      COMMON/FLAGS/RXFREE,TXFREE
      COMMON/MBUFS/MBUFS,MBFREE,LAST1,NEXT1
      COMMON/TRANSF/BUF,IBGBF,START,DATAX,DATAR
      COMMON/ICF/ICFLAG
      COMMON/ICFG/ICFG,STOPFG
      DATA YES/1HY/
C
C
C     INITIALISATION
C
      IF(.NOT.START)GOTO 235
C
C...I --- THE LOCAL DECISION UNIT NUMBER
C
      ICFLAG=0
      ICC=1
      DATDU(1)=6
      DATDU(2)=8
      DO 111 K=1,2
      IBGBF(K)=0
111   ICOUNT(K)=0
      DO 201 K=1,16
201   BUF(K)=0
      K0=0
      LQ=0
      I=1
      TFLAG=0
      START=.FALSE.
235   IF(TFLAG.EQ.1)GOTO 2080
      IF(TFLAG.EQ.2)GOTO 2090
C
C...TEST IF ANY DATA RECEIVED FROM IMIC
      IF(LAST1(I) .LT. 0)  GOTO 995
      JC=LAST1(I)+2
C
C...TEST IF NEW BLOCK OF DATA RECEIVED
C    AS A NEW BLOCK NUMBER IS TRANSFERRED
C    FROM THE IMIC WHEN TRANSFER TAKES PLACE
      IF(MBUFS(JC,I).GT.ICOUNT(I))GO TO 240
      GOTO 995
C
C...MBFREE(I) FLAG IS TO CONTROL THE UPDATING
C    OF THE MASTER BUFFER
240   MBFREE(I)=.FALSE.
C
C...SELECTS LATEST DATA BLOCK FROM MASTER BUFFER
      K1=1
110   K2=MBUFS(K1,I)-1
      K1=K2+1
```

```
          IF(K1 .LE. LAST1(I))  GOTO 110
          J=LAST1(I)
          K0=K0+1
          JC=J+2
          JS=J+4
          JF=JS+DATDU(I)
C
C...SETS LATEST DATA VALUES INTO TEMPORARY BUFFER
C    WHICH IS USED TO SEND DATA TO THE BACKGROUND
C    PROGRAM
          BUF(1)=ICC
          IF (I.EQ.2) GOTO 5011
          KJ=2
          GOTO 5012
5011      KJ=8
5012      DO 50 K=JS,JF
          BUF(KJ)=MBUFS(K,I)
          KJ=KJ+1
50        CONTINUE
C
          MBFREE(I)=.TRUE.
          ICOUNT(I)=MBUFS(JC,I)
          IF (I .EQ. 2) GOTO 2020
          I=2
          GOTO 235
C
2020      IF (LQ.EQ.1) GOTO 2030
C
C         SEND SYNCHRONISATION FLAG TO COORDINATOR
C
          NWORD=2
          TXB(2,1)=BUF(10)
          TXB(3,1)=BUF(11)
          TXB(2,2)=BUF(4)
          TXB(3,2)=BUF(5)
          DO 2010 I=1,NWORD
          IF (TXFREE(I)) GOTO 5025
          CALL PRINT(' TXB NOT FREE')
          GOTO 995
5025      TXB(1,I)=2*NWORD
          CALL TXFORD(TXB)
          TXFREE(I)=.FALSE.
2010      CONTINUE
          I=1
          LQ=1
          GOTO 995
C
2030      LQ=0
C         TYPE 9999,(BUF(K),K=1,15)
9999      FORMAT(15I5)
C
C         SET ALL THE NECESSARILY FLAGS TO SYNCHRONISE THE F/G & B/G PROGRA
C
          TFLAG=1
2080      IF(DATAX.EQ.2)GOTO 2070
          DATAX=1
          GOTO 995
2070      DATAX=0
C
C...SETS-UP TEMPORARY BUFFER TO RECEIVE
C    PARAMETER DATA FROM THE BACKGROUND PROGRAM
          DO 100 KM=1,3
          RECBF(KM)=0
100       CONTINUE
          TFLAG=2
2090      IF(DATAR.EQ.0)GOTO 995
```

```
        DATAR=2
        DO 2060 IJK=1,2
2060    RECBF(IJK)=IBGBF(IJK)
C
C...TXFREE(I) FLAG IS USED TO CONTROL
C   THE TRANSMISSION OF DATA TO THE IMIC
C   AS DATA MAY ONLY BE TRANSFERRED AFTER
C   THE IMIC HAS COMPLETED A TRANSFER
        DO 9998 I=1,2
        IF(TXFREE(I))  GOTO 125
        CALL PRINT(' TXBUF NOT FREE ')
        GOTO 995
125     CONTINUE
C
C...NWORDS....NUMBER OF PARAMETER DATA VALUES
        NWORDS=2
        DO 126 K=1,NWORDS
126     TXB(K+1,I)=RECBF(K)
        TXB(1,I)=2*NWORDS
C       TYPE 109,(TXB(J,I),J=2,3)
109     FORMAT(2I7)
        CALL TXFORD(TXB)
        TXFREE(I)=.FALSE.
9998    CONTINUE
C
        ICFLAG=1
        ICC=ICC+1
        I=1
        TFLAG=0
995     RETURN
        END
```

```
      SUBROUTINE ICIBGF(OP,ERROR,IDU,IRFUN)
C
C     THIS SUBROUTINE SENDS DATA FROM B/G PROGRAM TO THE INTECOLOR
C
C
      REAL*8 SEND,B(21)
      INTEGER*2 OP(21),IDU(2),ERROR(2),IRFUN
      BYTE BUFFX(21)
      EQUIVALENCE (BUFFX(1),SEND)
C
C
      N=8
      NN=21
      OP(17)=IDU(1)
      OP(18)=IDU(2)
      OP(19)=IRFUN
      OP(20)=ERROR(1)
      OP(21)=ERROR(2)
      DO 50 I=1,NN
      ENCODE(5,99,B(I)) OP(I)
50    CONTINUE
99    FORMAT(21I8)
      DO 200 I=1,NN
      SEND=B(I)
      CALL ARRAYX(N,BUFFX)
200   CONTINUE
      RETURN
      END
```

B2.3 Coordinator Software Listing

Background Routines for IPMLF

```
R LINK
DK:IPMLFB=DK:IPMLFB,IPLGET,DATEO,TIME/C
IPMLFF,LINKXD,RK,CON1,LKSET2/C
IPLFUN,TSXLIB,ICIPLF,GRAFLB,TIDY//
```

```
      PROGRAM IPMLFB
C
C     INTERACTION PREDICTION METHOD - LOCAL FEEDBACK
C
C     CO-ORDINATOR OPTIMISATION USING COMPLEX ALGORITHM
C
C
C
      INTEGER*2 BUF(17),RECBF(3),REPLY,STOPFG
      INTEGER*2 IBGBF(2),N,IC,ICOUNT,L,H,N1
      REAL*4 A(5),P(4,4),BGBF(2),STEP,ALPHA,BETA,GAMMA,FUN
      REAL*4 SS,T,VL,VH,SIZE,D,D1,TOL,X,B(4),MFUN
      LOGICAL*1 START,CR
      BYTE DATAX,DATAR
      COMMON/TRANSF/BUF,IBGBF,START,DATAX,DATAR
      COMMON/OUT/C11,C12,C21,C22,C23,RY11,RY21,RY22,ICOUNT,MFUN
      COMMON/ICFG/ICFG,STOPFG
C
C
C     DISABLE CONTROL 'C' TO AVOID CRASHING THE SYSTEM
C
      CALL SCCA(0)
C
      CALL PRINT(' INTECOLOR DISPLAY? KEY IN: 1(YES) ; 0(NO)')
      ACCEPT 1009,ICFG
1009  FORMAT(I2)
C
C     CALL ASSIGN(3,'IPMLF1.DAT',0,NEW,NC,1)
      CALL ASSIGN(4,'GRAPH.DAT',0,NEW,NC,1)
      CALL ASSIGN(6,'IPMLFB.DAT',0,NEW,NC,1)
      CALL ASSIGN(9,'CYGRA.DAT',0,NEW,NC,1)
      CALL ASSIGN(10,'PRINT.OUT',0,NEW,NC,1)
C
C
C     INITIALISATION OF I-MIC LINKS
C
      WRITE(6,928)
928   FORMAT(' INTERACTION PREDICTION METHOD WITH LOCAL FEEDBACK (SY)',
     1 //)
      CALL DATE0
      CALL TIME
      START=.TRUE.
      CALL LNKSET
      CALL TRMASC(30,85)
      CALL IPOKE('44,IPEEK('44).OR."100)
C
      A(1)=0.0
      N=2
      STEP=0.1
      IC=N+2
      ALPHA=1.3
      BETA=0.5
      GAMMA=2.0
      TOL=0.0002
      ICOUNT=1
      DATAR=0
      DATAX=0
      STOPFG=0
      WRITE(6,8)
8     FORMAT('ICOUNT V1      V2     P.MIN    P1C     P2C     P1H     P2H',/)
C     WRITE(3,804)
```

```
804     FORMAT(' ICOUNT    FMAX       FMID       FMIN   (FMAX-FMIN)   SSIZE')
C
C       FIRST ESTIMATE OF SIMPLEX VERTICE AND ITS FUNCTION VALUE
C
        CALL IPLGET(FUN,BGBF)
        IF (STOPFG.EQ.1) GOTO 143
        IF (FUN.LT.32.76) GOTO 15
        CALL PRINT(' ENTER ESTIMATE SET POINTS: V1,V2:- ')
        ACCEPT 16,BGBF(1),BGBF(2)
16      FORMAT(2F8.4)
        CALL CON1(BGBF,P)
        DO 17 L=1,N
        D=BGBF(L)*1000.0
        IBGBF(L)=INT(D)
17      CONTINUE
        CALL SYNCH
        CALL IPLGET(FUN,BGBF)
        IF (STOPFG.EQ.1) GOTO 143
15      P(N+1,1)=FUN
18      DO 10 I=1,N
        P(I,1)=BGBF(I)
10      CONTINUE
        WRITE(6,99) ICOUNT,BGBF(1),BGBF(2),FUN
99      FORMAT(I3,2F7.3,F8.4,32X,':GEN')
        COUNT=FLOAT(ICOUNT)
        DY1=0.05
        DY2=0.2
        WRITE(4,910) COUNT,BGBF(1),BGBF(2),FUN,DY1,DY2
        WRITE(9,95)COUNT,C11,C12,C21,C22,C23,RY11,RY21,RY22
95      FORMAT(F5.1,8F7.3)
        L=1
C
C       GENERATE REST OF SIMPLEX VERTICES AND THEIR FUNCTION VALUES
C
        SIZE=0.0
        DO 20 J=2,N+1
        DO 25 I=1,N
        P(I,J)=BGBF(I)
25      CONTINUE
        T=STEP
        IF (J.NE.2) GOTO 600
        D=3.0-T
        IF (BGBF(1).LT.D) GOTO 27
601     P(J-1,J)=BGBF(J-1)-T
        GOTO 26
600     D=1.846-T
        IF (BGBF(2).GT.D) GOTO 601
27      P(J-1,J)=BGBF(J-1)+T
26      IF (P(J-1,J).NE.BGBF(J-1)) GOTO 21
        T=T*10.0
        GOTO 27
21      SIZE=SIZE+ABS(T)
20      CONTINUE
23      SS=SIZE
        DO 30 J=1,N+1
        IF (J.EQ.L) GOTO 30
        DO 35 I=1,N
        BGBF(I)=P(I,J)
35      CONTINUE
        IF (ICOUNT .LT. 3) GOTO 606
        CALL CON1(BGBF,P)
606     DO 200 I=1,N
        D=BGBF(I)*1000.0
        IBGBF(I)=INT(D)
200     CONTINUE
        CALL SYNCH
```

```
        CALL IPLGET(FUN,BGBF)
        IF (STOPFG.EQ.1) GOTO 143
        ICOUNT=ICOUNT+1
        COUNT=COUNT+1.0
        F(N+1,J)=FUN
        IF (ICOUNT .GT. 3) GOTO 198
        WRITE(6,99) ICOUNT,BGBF(1),BGBF(2),FUN
        WRITE(4,910) COUNT,BGBF(1),BGBF(2),FUN,DY1,DY2
        WRITE(9,95)COUNT,C11,C12,C21,C22,C23,RY11,RY21,RY22
        GOTO 30
198     WRITE(6,2)ICOUNT,BGBF(1),BGBF(2),FUN,F(1,4),F(2,4),F(1,H),F(2,H)
2       FORMAT(I3,2F7.3,F8.4,4F7.3,4X,':GEN')
C       WRITE(3,1234) ICOUNT,VH,F(N+1,N1),VL,X,SS
        WRITE(4,910) COUNT,BGBF(1),BGBF(2),FUN,X,SS
910     FORMAT(F5.1,2F7.3,F8.4,2F10.5)
        WRITE(9,95)COUNT,C11,C12,C21,C22,C23,RY11,RY21,RY22
30      CONTINUE
C
C       PERFORM TESTS AND RE-ORDER FUNCTION VALUES
C
33      CONTINUE
        DO 40 I=1,N+1
40      B(I)=F(N+1,I)
        DO 41 I=1,N
        IJ=I+1
        DO 42 J=IJ,N+1
        IF (B(I).LE.B(J)) GOTO 42
        T=B(I)
        B(I)=B(J)
        B(J)=T
42      CONTINUE
41      CONTINUE
        DO 43 I=1,N+1
        DO 44 J=1,N+1
        IF (F(N+1,J).NE.B(I)) GOTO 44
        IF (I.EQ.1) GOTO 45
        IF (I.EQ.N) GOTO 46
        H=J
        GOTO 43
45      L=J
        GOTO 43
46      N1=J
44      CONTINUE
43      CONTINUE
        VL=F(N+1,L)
        VH=F(N+1,H)
        X=VH-VL
        TYPE 1234,ICOUNT,VH,F(N+1,N1),VL,X,SS
C       WRITE(3,1234) ICOUNT,VH,F(N+1,N1),VL,X,SS
1234    FORMAT(I5,5F10.5)
        WRITE(4,910) COUNT,BGBF(1),BGBF(2),FUN,X,SS
        IF (X.LE.TOL) GOTO 150
        IF (ICOUNT .LT. 100) GOTO 3456
        CALL TIME
        TYPE 3457
3457    FORMAT(' TYPE 0 (STOP) OR 1 (CONTINUE)')
        ACCEPT 3460,REPLY
3460    FORMAT(I3)
        IF (REPLY .EQ. 1) GOTO 3456
        GOTO 150
3456    D1=0.6*VH
        IF (VL.GT.D1) GOTO 49
        F(3,1)=F(3,L)
        BGBF(1)=F(1,L)
        BGBF(2)=F(2,L)
        GOTO 18
```

```
49      DO 50 I=1,N
        T=-F(I,H)
        DO 55 J=1,(N+1)
        T=T+F(I,J)
55      CONTINUE
        F(I,IC)=T/FLOAT(N)
50      CONTINUE
C
C       REFLECTION
C
        DO 60 I=1,N
        BGBF(I)=(1.0+ALPHA)*F(I,IC)-ALPHA*F(I,H)
60      CONTINUE
        CALL CON1(BGBF,F)
        DO 201 I=1,N
        D=BGBF(I)*1000.0
        IBGBF(I)=INT(D)
201     CONTINUE
        CALL SYNCH
        CALL IFLGET(FUN,BGBF)
        IF (STOPFG.EQ.1) GOTO 143
        ICOUNT=ICOUNT+1
        COUNT=COUNT+1.0
C       WRITE(3,1234) ICOUNT,VH,F(N+1,N1),VL,X,SS
        WRITE(4,910) COUNT,BGBF(1),BGBF(2),FUN,X,SS
        WRITE(6,3) ICOUNT,BGBF(1),BGBF(2),FUN,F(1,4),F(2,4),F(1,H),F(2,H)
3       FORMAT(I3,2F7.3,F8.4,4F7.3,4X,':REF')
        WRITE(9,95)COUNT,C11,C12,C21,C22,C23,RY11,RY21,RY22
        IF (FUN.LT.VL) GOTO 100
        IF (FUN.LT.F(N+1,N1)) GOTO 130
        IF (FUN.GE.VH) GOTO 80
        DO 70 I=1,N
        F(I,H)=BGBF(I)
70      CONTINUE
        F(N+1,H)=FUN
C
C       REDUCTION
C
80      DO 82 I=1,N
        BGBF(I)=(1.0-BETA)*F(I,H)+BETA*F(I,IC)
82      CONTINUE
        CALL CON1(BGBF,F)
        DO 202 I=1,N
        D=BGBF(I)*1000.0
        IBGBF(I)=INT(D)
202     CONTINUE
        CALL SYNCH
        CALL IFLGET(FUN,BGBF)
        IF (STOPFG.EQ.1) GOTO 143
        ICOUNT=ICOUNT+1.0
        COUNT=COUNT+1.0
C       WRITE(3,1234) ICOUNT,VH,F(N+1,N1),VL,X,SS
        WRITE(4,910) COUNT,BGBF(1),BGBF(2),FUN,X,SS
        WRITE(6,5) ICOUNT,BGBF(1),BGBF(2),FUN,F(1,4),F(2,4),F(1,H),F(2,H)
5       FORMAT(I3,2F7.3,F8.4,4F7.3,4X,':RED')
        WRITE(9,95)COUNT,C11,C12,C21,C22,C23,RY11,RY21,RY22
        IF (FUN.LT.F(N+1,H)) GOTO 130
C
C       CONTRACTION
C
        SIZE=0.0
        DO 90 J=1,N+1
        IF (J.EQ.L) GOTO 90
        DO 92 I=1,N
        F(I,J)=BETA*(F(I,J)-F(I,L))+F(I,L)
        SIZE=SIZE+ABS(F(I,J)-F(I,L))
```

```
92      CONTINUE
90      CONTINUE
        IF (SIZE.GE.0.0001) GOTO 446
        GOTO 150
446     IF (SIZE.LT.SS) GOTO 23
        CALL PRINT(' SIMPLEX FAILS TO CONTRACT')
        WRITE(6,444) SIZE,SS
444     FORMAT(//,2F12.5,///,'          SIMPLEX FAILS TO CONTRACT!',///)
        GOTO 23
C
C       EXTENSION
C
100     DO 102 I=1,N
        T=GAMMA*BGBF(I)+(1.0-GAMMA)*P(I,IC)
        P(I,IC)=BGBF(I)
        BGBF(I)=T
102     CONTINUE
        P(N+1,IC)=FUN
        CALL CON1(BGBF,P)
        DO 203 I=1,N
        D=BGBF(I)*1000.0
        IBGBF(I)=INT(D)
203     CONTINUE
        CALL SYNCH
        CALL IFLGET(FUN,BGBF)
        IF (STOPFG.EQ.1) GOTO 143
        ICOUNT=ICOUNT+1
        COUNT=COUNT+1.0
C       WRITE(3,1234) ICOUNT,YH,P(N+1,N1),YL,X,SS
        WRITE(4,910) COUNT,BGBF(1),BGBF(2),FUN,X,SS
        WRITE(6,4) ICOUNT,BGBF(1),BGBF(2),FUN,P(1,4),P(2,4),P(1,H),P(2,H)
4       FORMAT(I3,2F7.3,F8.4,4F7.3,4X,':EXT')
        WRITE(9,95)COUNT,C11,C12,C21,C22,C23,RY11,RY21,RY22
        IF (FUN.LT.P(N+1,IC)) GOTO 130
        DO 110 I=1,N
        BGBF(I)=P(I,IC)
110     CONTINUE
        FUN=P(N+1,IC)
130     DO 112 I=1,N
        P(I,H)=BGBF(I)
112     CONTINUE
        P(N+1,H)=FUN
        GOTO 33
150     DO 151 J=1,N
        BGBF(J)=P(J,L)
        D=BGBF(J)*1000.0
151     IBGBF(J)=INT(D)
        CALL SYNCH
        CALL IFLGET(FUN,BGBF)
        IF (STOPFG.EQ.1) GOTO 143
        ICOUNT=ICOUNT+1
        COUNT=COUNT+1.0
C       WRITE(3,1234) ICOUNT,YH,P(N+1,N1),YL,X,SS
        WRITE(4,910) COUNT,BGBF(1),BGBF(2),FUN,X,SS
        WRITE(6,6) ICOUNT,BGBF(1),BGBF(2),FUN,P(1,4),P(2,4),P(1,H),P(2,H)
6       FORMAT(I3,2F7.3,F8.4,4F7.3)
        WRITE(9,95)COUNT,C11,C12,C21,C22,C23,RY11,RY21,RY22
C
C       SOLUTION OUTPUT
C
143     WRITE(6,154) ICOUNT
154     FORMAT(//,' AFTER',I4,2X,'FUNCTION CALLS',/)
        WRITE(6,153) BGBF(1),BGBF(2)
153     FORMAT(' SET POINTS: V1=',F7.3,7X,'V2=',F7.3,/)
        WRITE(6,501) C11,C12
501     FORMAT(' C11= /,F7.3,5X,/C12= /,F7.3)
```

```
      WRITE(6,503) C21,C22,C23
503   FORMAT(' C21= ',F7.3,5X,'C22= ',F7.3,5X,'C23= ',F7.3)
      WRITE(6,504) RY11,RY21,RY22
504   FORMAT(' RY11=',F7.3,5X,'RY21=',F7.3,5X,'RY22=',F7.3,/)
      WRITE(6,506) MFUN
506   FORMAT(' FINAL MODEL FUNCTIONAL VALUE IS  ',F8.4,/)
      WRITE(6,505) FUN
505   FORMAT(' FINAL REAL  FUNCTIONAL VALUE IS  ',F8.4,//)
      SUBOPT=(FUN-5.92607)/0.0592607
      WRITE(6,507) SUBOPT
507   FORMAT(' SUBOPTIMALITY IS  ',F8.4,///)
C
C     SEND TERMINATION SIGNAL TO COORDINATOR
C
      IBGBF(1)=32766
      IBGBF(2)=32766
      CALL SYNCH
140   CALL TIME
      TYPE 678
678   FORMAT(1H ,' PLEASE STOP I-MICS AND THEN TYPE <CR>')
      ACCEPT 679,CR
679   FORMAT(A1)
      CALL TIDY2
      STOP
      END
C
C
C
      SUBROUTINE SYNCH
C
C     THIS SUBROUTINE SYNCHRONISES THE DATA TRANSFER BETWEEN
C     THE F/G & B/G PROGRAM
C
C
      INTEGER*2 BUF(17),IBGBF(2)
      LOGICAL*1 START
      BYTE DATAX,DATAR
      COMMON/TRANSF/BUF,IBGBF,START,DATAX,DATAR
C
C
      DATAR=1
20    IF(DATAR.EQ.2)GOTO 10
      GOTO 20
10    DATAR=0
      RETURN
```

```
      SUBROUTINE IPLGET(RFUN,GBF)
C
C     THIS SUBROUTINE GETS DATA FROM F/G PROGRAM (DECISION UNITS) TO THE
C     B/G PROGRAM (COORDINATOR) AND OUTPUT THE DATA ON INTECOLOR DISPLA
C
C
      INTEGER*2 BUF1(20),IBGBF(2),BUF(17),STOPFG,SHUT
      REAL*4 A1(3),BUFX(20),RFUN,GBF(2),RDU(2),RY11,RY21,RY22,MFUN
      REAL*4 DAIMIC(20)
      LOGICAL*1 START
      BYTE DATAX,DATAR
      COMMON/TRANSF/BUF,IBGBF,START,DATAX,DATAR
      COMMON/ICF/ICFLAG,DAIMIC
      COMMON/ICFG/ICFG,STOPFG
      COMMON/OUT/C11,C12,C21,C22,C23,RY11,RY21,RY22,ICOUNT,MFUN
      DATA YES/1HY/
C
C
      A1(1)=0.0
C     IF (.NOT.START) GOTO 4
C     DO 234 K=1,16
C234  DAIMIC(K)=FLOAT(BUF(K))/1000.0
C     CALL IPLFUN(DAIMIC,BUF1,RFUN,GBF)
4     CONTINUE
C
      IF (ICFG.EQ.0) GOTO 200
C
C     OUTPUT DATA TO INTECOLOR
C
      IF (ICFLAG.EQ.0) GOTO 200
      CALL ICIPLF(BUF1)
      ICFLAG=0
C
C     CHECK IF WANT TO SHUT DOWN THE SYSTEM
C
200   IVAL=ITTINR(0)
      IF (IVAL.LT.0) GOTO 210
      CALL CLEAR
      CALL PRINT(' ')
      CALL PRINT('           SYSTEM SHUT DOWN?')
      CALL PRINT(' TYPE IN:')
      CALL PRINT('           0 : NO')
      CALL PRINT('           1 : IMMEDIATELY')
      CALL PRINT('           2 : AFTER NEXT DATA SET')
      ACCEPT 90,SHUT
90    FORMAT(I2,/)
      IF (SHUT.EQ.0) GOTO 210
      STOPFG=1
      IF (SHUT.EQ.2) GOTO 210
      CALL TIDY2
      STOP
C
210   IF(DATAX.EQ.0)GOTO 4
C
C     SEND DATA TO B/G PROGRAM
C
      DATAX=2
      DO 5 K=1,16
5     BUFX(K)=(FLOAT(BUF(K)))/1000.0
C
      CALL IPLFUN(BUFX,BUF1,RFUN,GBF)
```

```
      RETURN
      END


      SUBROUTINE IPMLFF
C
C     THIS SUBROUTINE IS THE TIME CRITICAL F/G PROGRAM WHICH USED
C     TO COMMUNICATE SYNCHRONISELY WITH THE I-MIC'S THROUGH THE LINKS
C
C     USING INTERACTION PREDICTION METHOD WITH LOCAL FEEDBACK.
C
C
      INTEGER*2 BUF(17),RECBF(3),NEXT1(2),ICOUNT(2)
      INTEGER*2 IBGBF(2),UDF(2)
      INTEGER*2 LAST1(2),MBUFS(1000,2),YES,REPLY,TXB(128,2),DATDU(2)
      REAL*4 XY(20)
      BYTE DATAX,DATAR,TFLAG
      LOGICAL*1 TXFREE(2),MBFREE(2),START,RXFREE(2)
      COMMON/FLAGS/RXFREE,TXFREE
      COMMON/MBUFS/MBUFS,MBFREE,LAST1,NEXT1
      COMMON/TRANSF/BUF,IBGBF,START,DATAX,DATAR
      COMMON/ICF/ICFLAG
      COMMON/ICFG/ICFG,STOPFG
      DATA YES/1HY/
C
C
C     INITIALISATION
C
      IF(.NOT.START)GOTO 235
C
C...I --- THE LOCAL DECISION UNIT NUMBER
C
      ICFLAG=0
      ICC=1
      IDU=1
      DATDU(1)=6
      DATDU(2)=8
      DO 111 K=1,2
      IBGBF(K)=0
      UDF(K)=0
111   ICOUNT(K)=0
      DO 201 K=1,16
201   BUF(K)=0
      TFLAG=0
      I=1
      START=.FALSE.
235   IF(TFLAG.EQ.1)GOTO 2080
      IF(TFLAG.EQ.2)GOTO 2090
C
C...TEST IF ANY DATA RECEIVED FROM IMIC
      IF(LAST1(I) .LT. 0)  GOTO 995
      JC=LAST1(I)+2
C
C...TEST IF NEW BLOCK OF DATA RECEIVED
C   AS A NEW BLOCK NUMBER IS TRANSFERRED
C   FROM THE IMIC WHEN TRANSFER TAKES PLACE
      IF(MBUFS(JC,I).GT.ICOUNT(I))GO TO 240
      GOTO 995
C
C...MBFREE(I) FLAG IS TO CONTROL THE UPDATING
C   OF THE MASTER BUFFER
240   MBFREE(I)=.FALSE.
C
C...SELECTS LATEST DATA BLOCK FROM MASTER BUFFER
      K1=1
```

```
110     K2=MBUFS(K1,I)-1
        K1=K2+1
        IF(K1 .LE. LAST1(I))  GOTO 110
        J=LAST1(I)
        JS=J+4
        JF=JS+DATDU(I)
        UDF(I)=MBUFS(JC+1,I)
C
C...SETS LATEST DATA VALUES INTO TEMPORARY BUFFER
C   WHICH IS USED TO SEND DATA TO THE BACKGROUND
C   PROGRAM
C       BUF(1)=ICC
        IF (I.EQ.2) GOTO 5011
        KJ=2
        GOTO 5012
5011    KJ=8
5012    DO 50 K=JS,JF
        BUF(KJ)=MBUFS(K,I)
        KJ=KJ+1
50      CONTINUE
C       IF (I.EQ.1) GOTO 5678
        TYPE 151,(BUF(K),K=2,15)
151     FORMAT(14I5)
        BUF(16)=IDU
C
5678    MBFREE(I)=.TRUE.
        ICOUNT(I)=MBUFS(JC,I)
        IF (I. EQ. 2) GOTO 2020
        I=2
        GOTO 235
C
2020    IDU=IDU+1
C
C       CHECK IF THE DATA RECEIVED FROM THE DECISION UNITS
C       ARE CONVERGED SOLUTION
C
        IF (UDF(1).EQ.1 .AND. UDF(2).EQ.1) GOTO 2030
C
C       NOT CONVERGED SOLUTION, SEND SYNCHRONISATION FLAG TO COORDINATOR
C
        NWORD=3
        TXB(2,1)=BUF(9)
        TXB(3,1)=BUF(10)
        TXB(4,1)=UDF(2)
        TXB(2,2)=BUF(3)
        TXB(3,2)=BUF(4)
        TXB(4,2)=UDF(1)
C       TYPE 2101,TXB(2,1),TXB(3,1),TXB(4,1),TXB(2,2),TXB(3,2),TXB(4,2)
2101    FORMAT(6I8)
        DO 2010 I=1,2
        IF (TXFREE(I)) GOTO 5025
        CALL PRINT(' TXB NOT FREE')
        GOTO 995
5025    TXB(1,I)=2*NWORD
        CALL TXFORD(TXB)
        TXFREE(I)=.FALSE.
2010    CONTINUE
        I=1
        UDF(1)=0
        UDF(2)=0
        GOTO 995
C
C
C       STORE THE DATA OF THE CONVERGED DECISION UNIT SOLUTIONS INTO A
C       TEMPORARY BUFFER READY TO TRANSFER TO THE B/G PROGRAM
C
```

```
2030     LQ=0
C        TYPE 9999,(BUF(K),K=1,16)
9999     FORMAT(16I6)
C
C        SET ALL THE NECESSARILY FLAGS TO SYNCHRONISE THE F/G & B/G PROGRAI
C
         TFLAG=1
2080     IF(DATAX.EQ.2)GOTO 2070
         DATAX=1
         GOTO 995
2070     DATAX=0
C
C...SETS-UP TEMPORARY BUFFER TO RECEIVE
C   PARAMETER DATA FROM THE BACKGROUND PROGRAM
         DO 100 KM=1,3
         RECBF(KM)=0
100      CONTINUE
         TFLAG=2
2090     IF(DATAR.EQ.0)GOTO 995
         DATAR=2
         DO 2060 IJK=1,2
2060     RECBF(IJK)=IBGBF(IJK)
C
C...TXFREE(I) FLAG IS USED TO CONTROL
C   THE TRANSMISSION OF DATA TO THE IMIC
C   AS DATA MAY ONLY BE TRANSFERRED AFTER
C   THE IMIC HAS COMPLETED A TRANSFER
         DO 9998 I=1,2
2100     IF(TXFREE(I))  GOTO 125
         CALL PRINT(' TXBUF NOT FREE ')
         GOTO 995
125      CONTINUE
C
C...NWORDS....NUMBER OF PARAMETER DATA VALUES
         NWORDS=3
         DO 126 K=1,2
126      TXB(K+1,I)=RECBF(K)
         TXB(4,I)=UDF(I)
         TXB(1,I)=2*NWORDS
         TYPE 109,(TXB(J,I),J=2,4)
109      FORMAT(3I7)
         CALL TXFORD(TXB)
         TXFREE(I)=.FALSE.
9998     CONTINUE
C
         ICFLAG=1
         ICC=ICC+1
         I=1
         IDU=1
         TFLAG=0
         UDF(1)=0
         UDF(2)=0
995      RETURN
140      END
```

```
      SUBROUTINE IPLFUN(DAIMIC,BUF1,RFUN,GBF)
C
C     THIS SUBROUTINE IDENTIFIES AND DISPLAY THE DATA FROM THE IMICS AND
C     CALCULATES THE PERFORMANCE INDEX
C
C
      INTEGER*2 BUF1(20)
      REAL*4 A1(3),DAIMIC(20),RFUN,GBF(2),RDU(2),RY11,RY21,RY22,MFUN
      COMMON/OUT/C11,C12,C21,C22,C23,RY11,RY21,RY22,ICOUNT,MFUN
C
C
      GBF(1)=DAIMIC(2)
      GBF(2)=DAIMIC(8)
      S1=DAIMIC(7)
      S2=DAIMIC(15)
      C11=DAIMIC(3)
      C12=DAIMIC(4)
      C21=DAIMIC(9)
      C22=DAIMIC(10)
      C23=DAIMIC(11)
      RY11=DAIMIC(6)
      RY21=DAIMIC(14)
      RY22=2.3*C22-0.7*C23-1.1*RY11
      MFUN=DAIMIC(5)+DAIMIC(13)
C
C     DETERMINATION OF REAL PERFORMANCE
C
      RDU(1)=(RY11-1.0)**2+C11**2+C12**2
      RDU(2)=2.0*(RY21-2.0)**2+(RY22-3.0)**2+C21**2+C22**2+C23**2
      RFUN=RDU(1)+RDU(2)
C
C     SET UP A BUFFER OF DATA TO BE DISPLAY ON INTECOLOR
C
      DO 1000 K=1,16
1000  BUF1(K)=INT(DAIMIC(K)*1000.0)
      BUF1(17)=INT(RY22*1000.0)
      BUF1(18)=INT(RFUN*1000.0)
C
C     DISPLAY DATA FROM I-MICS
C
      TYPE 1,ICOUNT,GBF(1),GBF(2)
      WRITE(10,1) ICOUNT,GBF(1),GBF(2)
1     FORMAT(I3,' GLOBAL ITERATION :    V1=',F7.3,'    V2=',F7.3,/)
      TYPE 7,BUF1(16)-1,S1,S2
      WRITE(10,7) BUF1(16)-1,S1,S2
7     FORMAT(I3,' LOCAL  ITERATION :    S1=',F7.3,'    S2=',F7.3)
      TYPE 2,(DAIMIC(K),K=3,6)
      WRITE(10,2) (DAIMIC(K),K=3,6)
2     FORMAT(' DU1:  ',4F7.3)
      TYPE 3,(DAIMIC(K),K=9,14),RY22
      WRITE(10,3) (DAIMIC(K),K=9,14),RY22
3     FORMAT(' DU2:  ',7F7.3)
      TYPE 6,MFUN,RFUN
      WRITE(10,6) MFUN,RFUN
6     FORMAT(' MFI=',F7.3,'     RFI=',F7.3,/)
      RETURN
      END
```

B2.4 Coordinator Software Listing

Background Routines for IPMGF

```
R LINK
DK:IPMGFB=DK:IPMGFB,IPGGET,DATEO,TIME/C
IPMGFF,LINKXD,RK,CON1,LKSET2/C
TSXLIB,ICIPGF,GRAFLB,TIDY//
```

```
      PROGRAM IPMGFB
C
C     INTERACTION PREDICTION METHOD - GLOBAL FEEDBACK
C
C     CO-ORDINATOR OPTIMISATION USING COMPLEX ALGORITHM
C
C
C
      INTEGER*2 BUF(17),RECBF(3),REPLY,STOPFG
      INTEGER*2 IBGBF(2),N,IC,ICOUNT,L,H,N1
      REAL*4 A(5),P(4,4),BGBF(2),STEP,ALPHA,BETA,GAMMA,FUN
      REAL*4 SS,T,VL,VH,SIZE,D,D1,TOL,X,B(4),MFUN
      LOGICAL*1 START,CR
      BYTE DATAX,DATAR
      COMMON/TRANSF/BUF,IBGBF,START,DATAX,DATAR
      COMMON/OUT/C11,C12,C21,C22,C23,RY11,RY21,RY22,ICOUNT,MFUN
      COMMON/ICFG/ICFG,STOPFG
C
C
C     DISABLE CONTROL 'C' TO AVOID CRASHING THE SYSTEM
C
      CALL SCCA(0)
C
      CALL PRINT(' INTECOLOR DISPLAY? KEY IN: 1(YES) ; 0(NO)')
      ACCEPT 1009,ICFG
1009  FORMAT(I2)
C
      CALL ASSIGN(3,'IPMGF1.DAT',0,NEW,NC,1)
      CALL ASSIGN(4,'GRAPH.DAT',0,NEW,NC,1)
      CALL ASSIGN(6,'IPMGFB.DAT',0,NEW,NC,1)
      CALL ASSIGN(9,'CUGRA.DAT',0,NEW,NC,1)
C
C
C     INITIALISATION OF I-MIC LINKS
C
      WRITE(6,928)
928   FORMAT(' INTERACTION PREDICTION METHOD WITH GLOBAL FEEDBACK (SY)',
     1 //)
      CALL DATEO
      CALL TIME
      START=.TRUE.
      CALL LNKSET
      CALL TRMASC(30,85)
      CALL IPOKE('44,IPEEK('44).OR.'100)
C
      A(1)=0.0
      N=2
      STEP=0.1
      IC=N+2
      ALPHA=1.3
      BETA=0.5
      GAMMA=2.0
      TOL=0.0002
      ICOUNT=1
      DATAR=0
      DATAX=0
      STOPFG=0
      WRITE(6,8)
8     FORMAT('ICOUNT V1      V2     P.MIN    P1C     P2C     P1H     P2H',/)
      WRITE(3,804)
804   FORMAT(' ICOUNT    PMAX      PMIN       PMIN  (PMAX-PMIN). SSIZE')
```

```
C
C       FIRST ESTIMATE OF SIMPLEX VERTICE AND ITS FUNCTION VALUE
C
        CALL IFGGET(FUN,BGBF)
        IF (STOPFG.EQ.1) GOTO 143
        IF (FUN.LT.32.76) GOTO 15
        CALL PRINT(' ENTER ESTIMATE SET POINTS: V1,V2:- ')
        ACCEPT 16,BGBF(1),BGBF(2)
16      FORMAT(2F8.4)
        CALL CON1(BGBF,P)
        DO 17 L=1,N
        D=BGBF(L)*1000.0
        IBGBF(L)=INT(D)
17      CONTINUE
        CALL SYNCH
        CALL IFGGET(FUN,BGBF)
        IF (STOPFG.EQ.1) GOTO 143
15      P(N+1,1)=FUN
18      DO 10 I=1,N
        P(I,1)=BGBF(I)
10      CONTINUE
        WRITE(6,99) ICOUNT,BGBF(1),BGBF(2),FUN
99      FORMAT(I3,2F7.3,F8.4,32X,':GEN')
        COUNT=FLOAT(ICOUNT)
        DY1=0.05
        DY2=0.2
        WRITE(4,910) COUNT,BGBF(1),BGBF(2),FUN,DY1,DY2
        WRITE(9,95)COUNT,C11,C12,C21,C22,C23,RY11,RY21,RY22
95      FORMAT(F5.1,8F7.3)
        L=1
C
C       GENERATE REST OF SIMPLEX VERTICES AND THEIR FUNCTION VALUES
C
        SIZE=0.0
        DO 20 J=2,N+1
        DO 25 I=1,N
        P(I,J)=BGBF(I)
25      CONTINUE
        T=STEP
        IF (J.NE.2) GOTO 600
        D=3.0-T
        IF (BGBF(1).LT.D) GOTO 27
601     P(J-1,J)=BGBF(J-1)-T
        GOTO 26
600     D=1.846-T
        IF (BGBF(2).GT.D) GOTO 601
27      P(J-1,J)=BGBF(J-1)+T
26      IF (P(J-1,J).NE.BGBF(J-1)) GOTO 21
        T=T*10.0
        GOTO 27
21      SIZE=SIZE+ABS(T)
20      CONTINUE
23      SS=SIZE
        DO 30 J=1,N+1
        IF (J.EQ.L) GOTO 30
        DO 35 I=1,N
        BGBF(I)=P(I,J)
35      CONTINUE
        IF (ICOUNT .LT. 3) GOTO 606
        CALL CON1(BGBF,P)
606     DO 200 I=1,N
        D=BGBF(I)*1000.0
        IBGBF(I)=INT(D)
200     CONTINUE
        CALL SYNCH
        CALL IFGGET(FUN,BGBF)
```

```
      IF (STOPFG.EQ.1) GOTO 143
      ICOUNT=ICOUNT+1
      COUNT=COUNT+1.0
      F(N+1,J)=FUN
      IF (ICOUNT .GT. 3) GOTO 198
      WRITE(6,99) ICOUNT,BGBF(1),BGBF(2),FUN
      WRITE(4,910) COUNT,BGBF(1),BGBF(2),FUN,DY1,DY2
      WRITE(9,95)COUNT,C11,C12,C21,C22,C23,RY11,RY21,RY22
      GOTO 30
198   WRITE(6,2)ICOUNT,BGBF(1),BGBF(2),FUN,F(1,4),F(2,4),F(1,H),F(2,H)
2     FORMAT(I3,2F7.3,F8.4,4F7.3,4X,':GEN')
      WRITE(3,1234) ICOUNT,VH,F(N+1,N1),VL,X,SS
      WRITE(4,910) COUNT,BGBF(1),BGBF(2),FUN,X,SS
910   FORMAT(F5.1,2F7.3,F8.4,2F10.5)
      WRITE(9,95)COUNT,C11,C12,C21,C22,C23,RY11,RY21,RY22
30    CONTINUE
C
C     PERFORM TESTS AND RE-ORDER FUNCTION VALUES
C
33    CONTINUE
      DO 40 I=1,N+1
40    B(I)=F(N+1,I)
      DO 41 I=1,N
      IJ=I+1
      DO 42 J=IJ,N+1
      IF (B(I).LE.B(J)) GOTO 42
      T=B(I)
      B(I)=B(J)
      B(J)=T
42    CONTINUE
41    CONTINUE
      DO 43 I=1,N+1
      DO 44 J=1,N+1
      IF (F(N+1,J).NE.B(I)) GOTO 44
      IF (I.EQ.1) GOTO 45
      IF (I.EQ.N) GOTO 46
      H=J
      GOTO 43
45    L=J
      GOTO 43
46    N1=J
44    CONTINUE
43    CONTINUE
      VL=F(N+1,L)
      VH=F(N+1,H)
      X=VH-VL
      TYPE 1234,ICOUNT,VH,F(N+1,N1),VL,X,SS
      WRITE(3,1234) ICOUNT,VH,F(N+1,N1),VL,X,SS
1234  FORMAT(I5,5F10.5)
      WRITE(4,910) COUNT,BGBF(1),BGBF(2),FUN,X,SS
      IF (X.LE.TOL) GOTO 150
      IF (ICOUNT .LT. 100) GOTO 3456
      CALL TIME
      TYPE 3457
3457  FORMAT(' TYPE 0 (STOP) OR 1 (CONTINUE)')
      ACCEPT 3460,REPLY
3460  FORMAT(I3)
      IF (REPLY .EQ. 1) GOTO 3456
      GOTO 150
3456  D1=0.6*VH
      IF (VL.GT.D1) GOTO 49
      F(3,1)=F(3,L)
      BGBF(1)=F(1,L)
      BGBF(2)=F(2,L)
      GOTO 18
49    DO.50 I=1,N
```

```
      T=-F(I,H)
      DO 55 J=1,(N+1)
      T=T+F(I,J)
55    CONTINUE
      F(I,IC)=T/FLOAT(N)
50    CONTINUE
C
C     REFLECTION
C
      DO 60 I=1,N
      BGBF(I)=(1.0+ALPHA)*F(I,IC)-ALPHA*F(I,H)
60    CONTINUE
      CALL CON1(BGBF,F)
      DO 201 I=1,N
      D=BGBF(I)*1000.0
      IBGBF(I)=INT(D)
201   CONTINUE
      CALL SYNCH
      CALL IFGGET(FUN,BGBF)
      IF (STOPFG.EQ.1) GOTO 143
      ICOUNT=ICOUNT+1
      COUNT=COUNT+1.0
      WRITE(3,1234) ICOUNT,VH,F(N+1,N1),VL,X,SS
      WRITE(4,910) COUNT,BGBF(1),BGBF(2),FUN,X,SS
      WRITE(6,3) ICOUNT,BGBF(1),BGBF(2),FUN,F(1,4),F(2,4),F(1,H),F(2,H)
3     FORMAT(I3,2F7.3,F8.4,4F7.3,4X,':REF')
      WRITE(9,95)COUNT,C11,C12,C21,C22,C23,RY11,RY21,RY22
      IF (FUN.LT.VL) GOTO 100
      IF (FUN.LT.F(N+1,N1)) GOTO 130
      IF (FUN.GE.VH) GOTO 80
      DO 70 I=1,N
      F(I,H)=BGBF(I)
70    CONTINUE
      F(N+1,H)=FUN
C
C     REDUCTION
C
80    DO 82 I=1,N
      BGBF(I)=(1.0-BETA)*F(I,H)+BETA*F(I,IC)
82    CONTINUE
      CALL CON1(BGBF,F)
      DO 202 I=1,N
      D=BGBF(I)*1000.0
      IBGBF(I)=INT(D)
202   CONTINUE
      CALL SYNCH
      CALL IFGGET(FUN,BGBF)
      IF (STOPFG.EQ.1) GOTO 143
      ICOUNT=ICOUNT+1.0
      COUNT=COUNT+1.0
      WRITE(3,1234) ICOUNT,VH,F(N+1,N1),VL,X,SS
      WRITE(4,910) COUNT,BGBF(1),BGBF(2),FUN,X,SS
      WRITE(6,5) ICOUNT,BGBF(1),BGBF(2),FUN,F(1,4),F(2,4),F(1,H),F(2,H)
5     FORMAT(I3,2F7.3,F8.4,4F7.3,4X,':RED')
      WRITE(9,95)COUNT,C11,C12,C21,C22,C23,RY11,RY21,RY22
      IF (FUN.LT.F(N+1,H)) GOTO 130
C
C     CONTRACTION
C
      SIZE=0.0
      DO 90 J=1,N+1
      IF (J.EQ.L) GOTO 90
      DO 92 I=1,N
      F(I,J)=BETA*(F(I,J)-F(I,L))+F(I,L)
      SIZE=SIZE+ABS(F(I,J)-F(I,L))
92    CONTINUE
```

```
90      CONTINUE
        IF (SIZE.GE.0.0001) GOTO 446
        GOTO 150
446     IF (SIZE.LT.SS) GOTO 23
        CALL PRINT(' SIMPLEX FAILS TO CONTRACT')
        WRITE(6,444) SIZE,SS
444     FORMAT(//,2F12.5,///,'          SIMPLEX FAILS TO CONTRACT!',///)
        GOTO 23
C
C       EXTENSION
C
100     DO 102 I=1,N
        T=GAMMA*BGBF(I)+(1.0-GAMMA)*P(I,IC)
        P(I,IC)=BGBF(I)
        BGBF(I)=T
102     CONTINUE
        P(N+1,IC)=FUN
        CALL CON1(BGBF,P)
        DO 203 I=1,N
        D=BGBF(I)*1000.0
        IBGBF(I)=INT(D)
203     CONTINUE
        CALL SYNCH
        CALL IPGGET(FUN,BGBF)
        IF (STOPFG.EQ.1) GOTO 143
        ICOUNT=ICOUNT+1
        COUNT=COUNT+1.0
        WRITE(3,1234) ICOUNT,VH,P(N+1,N1),VL,X,SS
        WRITE(4,910) COUNT,BGBF(1),BGBF(2),FUN,X,SS
        WRITE(6,4) ICOUNT,BGBF(1),BGBF(2),FUN,P(1,4),P(2,4),P(1,H),P(2,H)
4       FORMAT(I3,2F7.3,F8.4,4F7.3,4X,':EXT')
        WRITE(9,95)COUNT,C11,C12,C21,C22,C23,RY11,RY21,RY22
        IF (FUN.LT.P(N+1,IC)) GOTO 130
        DO 110 I=1,N
        BGBF(I)=P(I,IC)
110     CONTINUE
        FUN=P(N+1,IC)
130     DO 112 I=1,N
        P(I,H)=BGBF(I)
112     CONTINUE
        P(N+1,H)=FUN
        GOTO 33
150     DO 151 J=1,N
        BGBF(J)=P(J,L)
        D=BGBF(J)*1000.0
151     IBGBF(J)=INT(D)
        CALL SYNCH
        CALL IPGGET(FUN,BGBF)
        IF (STOPFG.EQ.1) GOTO 143
        ICOUNT=ICOUNT+1
        COUNT=COUNT+1.0
        WRITE(3,1234) ICOUNT,VH,P(N+1,N1),VL,X,SS
        WRITE(4,910) COUNT,BGBF(1),BGBF(2),FUN,X,SS
        WRITE(6,6) ICOUNT,BGBF(1),BGBF(2),FUN,P(1,4),P(2,4),P(1,H),P(2,H
6       FORMAT(I3,2F7.3,F8.4,4F7.3)
        WRITE(9,95)COUNT,C11,C12,C21,C22,C23,RY11,RY21,RY22
C
C       SOLUTION OUTPUT
C
143     WRITE(6,154) ICOUNT
154     FORMAT(//,' AFTER',I4,2X,'FUNCTION CALLS',/)
        WRITE(6,153) BGBF(1),BGBF(2)
153     FORMAT(' SET POINTS: V1=',F7.3,7X,'V2=',F7.3,/)
        WRITE(6,501) C11,C12
501     FORMAT(' C11= ',F7.3,5X,'C12= ',F7.3)
        WRITE(6,503) C21,C22,C23
```

```
503     FORMAT(' C21= ',F7.3,5X,'C22= ',F7.3,5X,'C23= ',F7.3)
        WRITE(6,504) RY11,RY21,RY22
504     FORMAT(' RY11=',F7.3,5X,'RY21=',F7.3,5X,'RY22=',F7.3,/)
        WRITE(6,506) MFUN
506     FORMAT(' FINAL MODEL FUNCTIONAL VALUE IS  ',F8.4)
        WRITE(6,505) FUN
505     FORMAT(' FINAL REAL FUNCTIONAL VALUE IS   ',F8.4,//)
C
C       SEND TERMINATION SIGNAL TO COORDINATOR
C
        IBGBF(1)=32766
        IBGBF(2)=32766
        CALL SYNCH
140     CALL TIME
        TYPE 678
678     FORMAT(1H ,' PLEASE STOP I-MICS AND THEN TYPE <CR>')
        ACCEPT 679,CR
679     FORMAT(A1)
        CALL TIDY2
        STOP
        END
C
C
C
        SUBROUTINE SYNCH
C
C       THIS SUBROUTINE SYNCHRONISES THE DATA TRANSFER BETWEEN
C       THE F/G & B/G PROGRAM
C
C
        INTEGER*2 BUF(17),IBGBF(2)
        LOGICAL*1 START
        BYTE DATAX,DATAR
        COMMON/TRANSF/BUF,IBGBF,START,DATAX,DATAR
C
C
        DATAR=1
20      IF(DATAR.EQ.2)GOTO 10
        GOTO 20
10      DATAR=0
        RETURN
        END
```

```
      SUBROUTINE IPGGET(RFUN,GBF)
C
C     THIS SUBROUTINE GETS DATA FROM F/G PROGRAM (DECISION UNITS) TO THE
C     B/G PROGRAM (COORDINATOR) AND OUTPUT THE DATA ON INTECOLOR DISPLAY
C
C
      INTEGER*2 BUF1(20),IBGBF(2),BUF(17),STOPFG,SHUT
      REAL*4 A1(3),BUFX(17),RFUN,GBF(2),RDU(2),RY11,RY21,RY22,MFUN
      LOGICAL*1 START
      BYTE DATAX,DATAR
      COMMON/TRANSF/BUF,IBGBF,START,DATAX,DATAR
      COMMON/ICF/ICFLAG
      COMMON/ICFG/ICFG,STOPFG
      COMMON/OUT/C11,C12,C21,C22,C23,RY11,RY21,RY22,ICOUNT,MFUN
      DATA YES/1HY/
C
C
      A1(1)=0.0
4     CONTINUE
C
      IF (ICFG.EQ.0) GOTO 200
C
C     OUTPUT DATA TO INTECOLOR
C
      IF (ICFLAG.EQ.0) GOTO 200
      CALL ICIPGF(BUF1)
      ICFLAG=0
C
C     CHECK IF WANT TO SHUT DOWN THE SYSTEM
C
200   IVAL=ITTINR(0)
      IF (IVAL.LT.0) GOTO 210
      CALL CLEAR
      CALL PRINT(' ')
      CALL PRINT('           SYSTEM SHUT DOWN?')
      CALL PRINT(' TYPE IN:')
      CALL PRINT('           0 : NO')
      CALL PRINT('           1 : IMMEDIATELY')
      CALL PRINT('           2 : AFTER NEXT DATA SET')
      ACCEPT 90,SHUT
90    FORMAT(I2,/)
      IF (SHUT.EQ.0) GOTO 210
      STOPFG=1
      IF (SHUT.EQ.2) GOTO 210
      CALL TIDY2
      STOP
C
210   IF(DATAX.EQ.0)GOTO 4
C
C     SEND DATA TO B/G PROGRAM
C
      DATAX=2
      DO 5 K=1,12
5     BUFX(K)=(FLOAT(BUF(K)))/1000.0
C
      GBF(1)=BUFX(2)
      GBF(2)=BUFX(7)
      C11=BUFX(3)
      C12=BUFX(4)
      C21=BUFX(8)
      C22=BUFX(9)
```

```
      C23=BUFX(10)
      RY11=BUFX(6)
      RY21=BUFX(12)
      RY22=2.3*C22-0.7*C23-1.1*RY11
      MFUN=BUFX(5)+BUFX(11)
C
C     DETERMINATION OF REAL PERFORMANCE
C
      RDU(1)=(RY11-1.0)**2+C11**2+C12**2
      RDU(2)=2.0*(RY21-2.0)**2+(RY22-3.0)**2+C21**2+C22**2+C23**2
      RFUN=RDU(1)+RDU(2)
C
C     SET UP A BUFFER OF DATA TO BE DISPLAY ON INTECOLOR
C
      DO 1000 K=1,12
1000  BUF1(K)=BUF(K)
      BUF1(13)=INT(RY22*1000.0)
      BUF1(14)=INT(RFUN*1000.0)
C
C     DISPLAY DATA FROM I-MICS
C
      TYPE 1,ICOUNT,GBF(1),GBF(2)
1     FORMAT(I3,' ITERATION :    V1=',F7.3,'    V2=',F7.3)
      TYPE 2,(BUFX(K),K=3,6)
2     FORMAT(' DU1:  ',4F7.3)
      TYPE 3,(BUFX(K),K=8,12),RY22
3     FORMAT(' DU2:  ',6F7.3)
      TYPE 6,MFUN,RFUN
6     FORMAT(' MFI=',F7.3,'     RFI=',F7.3,/)
      RETURN
      END
```

```
C
      SUBROUTINE IPMGFF
C
C     THIS SUBROUTINE IS THE TIME CRITICAL F/G PROGRAM WHICH USED
C     TO COMMUNICATE SYNCHRONISELY WITH THE I-MIC'S THROUGH THE LINKS
C
C     USING INTERACTION PREDICTION METHOD WITH GLOBAL FEEDBACK.
C
C
      INTEGER*2 BUF(17),RECBF(3),NEXT1(2),ICOUNT(2)
      INTEGER*2 IBGBF(2)
      INTEGER*2 LAST1(2),MBUFS(1000,2),YES,REPLY,TXB(128,2),DATDU(2)
      BYTE DATAX,DATAR,TFLAG
      LOGICAL*1 TXFREE(2),MBFREE(2),START,RXFREE(2)
      COMMON/FLAGS/RXFREE,TXFREE
      COMMON/MBUFS/MBUFS,MBFREE,LAST1,NEXT1
      COMMON/TRANSF/BUF,IBGBF,START,DATAX,DATAR
      COMMON/ICF/ICFLAG
      COMMON/ICFG/ICFG,STOPFG
      DATA YES/1HY/
C
C
C     INITIALISATION
C
      IF(.NOT.START)GOTO 235
C
C...I --- THE LOCAL DECISION UNIT NUMBER
C
      ICFLAG=0
      ICC=1
      DATDU(1)=5
      DATDU(2)=6
      DO 111 K=1,2
      IBGBF(K)=0
111   ICOUNT(K)=0
      DO 201 K=1,12
201   BUF(K)=0
      KO=0
      LQ=0
      TFLAG=0
      I=1
      START=.FALSE.
235   IF(TFLAG.EQ.1)GOTO 2080
      IF(TFLAG.EQ.2)GOTO 2090
C
C...TEST IF ANY DATA RECEIVED FROM IMIC
      IF(LAST1(I) .LT. 0)  GOTO 995
      JC=LAST1(I)+2
C
C...TEST IF NEW BLOCK OF DATA RECEIVED
C    AS A NEW BLOCK NUMBER IS TRANSFERRED
C    FROM THE IMIC WHEN TRANSFER TAKES PLACE
      IF(MBUFS(JC,I).GT.ICOUNT(I))GO TO 240
      GOTO 995
C
C...MBFREE(I) FLAG IS TO CONTROL THE UPDATING
C    OF THE MASTER BUFFER
240   MBFREE(I)=.FALSE.
C
C...SELECTS LATEST DATA BLOCK FROM MASTER BUFFER
      K1=1
```

```
110     K2=MBUFS(K1,I)-1
        K1=K2+1
        IF(K1 .LE. LAST1(I))  GOTO 110
        J=LAST1(I)
        K0=K0+1
        JC=J+2
        JS=J+4
        JF=JS+DATDU(I)
C
C...SETS LATEST DATA VALUES INTO TEMPORARY BUFFER
C   WHICH IS USED TO SEND DATA TO THE BACKGROUND
C   PROGRAM
        BUF(1)=ICC
        IF (I.EQ.2) GOTO 5011
        KJ=2
        GOTO 5012
5011    KJ=7
5012    DO 50 K=JS,JF
        BUF(KJ)=MBUFS(K,I)
        KJ=KJ+1
50      CONTINUE
C
        MBFREE(I)=.TRUE.
        ICOUNT(I)=MBUFS(JC,I)
        IF (I. EQ. 2) GOTO 2020
        I=2
        GOTO 235
C
2020    IF (LQ.EQ.1) GOTO 2030
C

C       SEND SYNCHRONISATION FLAG TO COORDINATOR
C
        NWORD=2
        TXB(2,1)=BUF(8)
        TXB(3,1)=BUF(9)
        TXB(2,2)=BUF(3)
        TXB(3,2)=BUF(4)
        DO 2010 I=1,NWORD
        IF (TXFREE(I)) GOTO 5025
        CALL PRINT(' TXB NOT FREE')
        GOTO 995
5025    TXB(1,I)=2*NWORD
        CALL TXFORD(TXB)
        TXFREE(I)=.FALSE.
2010    CONTINUE
        I=1
        LQ=1
        GOTO 995
C
C
C       STORE THE DATA OF THE CONVERGED DECISION UNIT SOLUTIONS INTO A
C       TEMPORARY BUFFER READY TO TRANSFER TO THE B/G PROGRAM
C
2030    LQ=0
C       TYPE 9999,(BUF(K),K=1,12)
9999    FORMAT(12I6)
C
C       SET ALL THE NECESSARILY FLAGS TO SYNCHRONISE THE F/G & B/G PROGRA
C
        TFLAG=1
2080    IF(DATAX.EQ.2)GOTO 2070
        DATAX=1
        GOTO 995
2070    DATAX=0
C
```

```
C...SETS-UP TEMPORARY BUFFER TO RECEIVE
C    PARAMETER DATA FROM THE BACKGROUND PROGRAM
          DO 100 KM=1,3
          RECBF(KM)=0
100       CONTINUE
          TFLAG=2
2090      IF(DATAR.EQ.0)GOTO 995
          DATAR=2
          DO 2060 IJK=1,2
2060      RECBF(IJK)=IBGBF(IJK)
C
C...TXFREE(I) FLAG IS USED TO CONTROL
C    THE TRANSMISSION OF DATA TO THE IMIC
C    AS DATA MAY ONLY BE TRANSFERRED AFTER
C    THE IMIC HAS COMPLETED A TRANSFER
          DO 9998 I=1,2
2100      IF(TXFREE(I))  GOTO 125
          CALL PRINT(' TXBUF NOT FREE ')
          GOTO 995
125       CONTINUE
C
C...NWORDS....NUMBER OF PARAMETER DATA VALUES
          NWORDS=2
          DO 126 K=1,NWORDS
126       TXB(K+1,I)=RECBF(K)
          TXB(1,I)=2*NWORDS
C         TYPE 109,(TXB(J,I),J=2,3)
109       FORMAT(2I7)
          CALL TXFORD(TXB)
          TXFREE(I)=.FALSE.
9998      CONTINUE
C
          ICFLAG=1
          ICC=ICC+1
          I=1
          TFLAG=0
995       RETURN
140       END
```

```
      SUBROUTINE CON1(XGBF,P)
C
      REAL*4 A2(3),D1,D2,XGBF(2),P(4,4),E1,E2,E3
C
780   IF (XGBF(1).LE.3.0.AND.XGBF(1).GE.0.0) GOTO 800
      GOTO 1000
800   D1=XGBF(1)-2.0
      D2=(XGBF(1)+1.8)/2.6
      IF (XGBF(2).GE.0.0.AND.XGBF(2).GE.D1.AND.XGBF(2).LE.D2) GOTO 880
1000  XGBF(1)=(P(1,4)+XGBF(1))/2.0
      XGBF(2)=(P(2,4)+XGBF(2))/2.0
      CALL PRINT(' CONSTRAIN VIOLATION!!')
      E1=P(1,4)-XGBF(1)
      E2=P(2,4)-XGBF(2)
      IF (ABS(E1).LE.0.001 .OR. ABS(E2).LE.0.001) GOTO 900
      GOTO 780
900   E3=0.1
      IF (P(1,4).EQ.0.0) GOTO 910
      IF (P(1,4).EQ.3.0) GOTO 920
      XGBF(1)=P(1,4)
      XGBF(2)=P(2,4)
      GOTO 880
910   XGBF(1)=P(1,4)+E3
      XGBF(2)=P(2,4)+E3
      GOTO 880
920   E3=-E3
      GOTO 910
880   RETURN
      END
```

```
      SUBROUTINE ICIFGF(OF)
C
C     THIS SUBROUTINE SENDS DATA FROM B/G PROGRAM TO THE INTECOLOR
C
C
      REAL*8 SEND,B(20)
      INTEGER*2 OF(20)
      BYTE BUFFX(20)
      EQUIVALENCE (BUFFX(1),SEND)
C
C
      N=8
      NN=14
      DO 50 I=1,NN
      ENCODE(5,99,B(I)) OF(I)
50    CONTINUE
99    FORMAT(14I8)
      DO 200 I=1,NN
      SEND=B(I)
      CALL ARRAYX(N,BUFFX)
200   CONTINUE
      RETURN
      END
```

C1. Local Decision Units (I-MICs) Software Listing

for

Interaction Prediction Method with Global Feedback

```
400 ;PROGRAM 071283
405 P.'INTERACTION PREDICATION METHOD WITH GLOBAL FEEDBACK'
410 P.
415 P.'LOCAL DECISION UNIT 1'
420 P.
425 INP.'DATE'A
430 INP.'TIME',B
435 P.
440 P.
445 P.'  N     V1     V2    C11    C12    P1    Y11*'
446 P.'*****************************************'
500 D.(3,2,4,4,5,3,15,21,21,21,7,1,10,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,20,10)
501 ;
506 ;INITIALISATION
508 ;INP'V1'X,'V2'Y
510 E=1 : Q=1
512 F.I=1 TO 3
513 A(I)=0,F(I)=0 : N.I
515 F.I=1 TO 4
516 C(I)=0,D(I)=0 : N.I
518 B(1)=0,B(2)=0
519 Z(0)=0,Z(1)=0,Z(2)=0
521 ;
522 ;DECISION UNIT 1 OPTIMISATION ROUTINE
525 ; DETERMINE C11 MAX(B(1))  &  C11 MIN(B(2))
530 A(1)=X-2*Y+1000
540 A(2)=800+X-26*Y/10
550 A(3)=X-2*Y-1000
560 IF A(1)<1000 G.590
570 B(1)=1000
580 GOTO 600
590 B(1)=A(1)
600 IF A(2)>B(1) GOTO 630
610 B(1)=A(2)
630 IF A(3)<-1000 GOTO 660
640 B(2)=A(3)
650 G.700
660 B(2)=-1000
670 G.1500
675 ;
700 ; NEW OPTIMISATION ROUTINE (2-FEB-84)
705 C(1)=X/2-Y
710 IF C(1)>B(1) G.725
715 IF C(1)<B(2) G.730
720 G.750
725 C(1)=B(1) : G.750
730 C(1)=B(2)
750 C(2)=C(1)+2*Y-X
760 C(3)=(X-1000)*(X-1000)/1000+C(1)*C(1)/1000+C(2)*C(2)/1000
770 G.1720
780 ;
1500 G(1)=-1000, G(2)=1000, G(3)=A(2)
1510 FOR I=1 TO 3
1520 IF G(I)>B(1) G.1580
1530 IF G(I)<B(2) G.1580
1540 D(1)=(X-1000)*(X-1000)/1000
1550 D(2)=(G(I)+2*Y-X)*(G(I)+2*Y-X)/1000
1560 F(I)=D(1)+D(2)+G(I)*G(I)/1000+1
```

```
1570 GOTO 1590
1580 F(I)=32767,G(I)=F(I)
1590 N.I
1597 F(0)=32767, G(0)=F(0)
1600 F.I=1 TO 3
1610 IF F(I-1)>F(I) GOTO 1640
1620 F(I)=F(I-1),G(I)=G(I-1)
1630 GOTO 1650
1640 F(I)=F(I),G(I)=G(I)
1650 NEXT I
1660 C(1)=G(3)
1662 IF C(1)=32767 G.1672
1670 C(2)=C(1)+2*Y-X
1671 G.1675
1672 C(2)=32767
1675 C(3)=F(3)
1685 G.1720
1700 P.
1705 P.'C11 =',C(1),'     C12 =',C(2)
1710 P.'P1 =',C(3)
1715 P.
1717 ;
1718 ;SYNCHRONISATION ROUTINE
1720 EI
1721 W.(50)
1725 COS.5000
1730 IF Z(0)>0 G.1745
1735 W.(10)
1740 G.1725
1745 IF Z(1)#32766 G.1755
1750 S.
1755 Q=Q+1
1760 Z(0)=0
1765 W=(6*C(2)/10-14*C(1)/10+198*Z(2)/100-234*Z(1)/100)*100/98
1780 ;
1790 ;SEND CONTROLS TO TR48 AND TAKE REAL MEASUREMENTS
1800 APO.(H.3E04,C(1))
1805 APO.(H.3E06,C(2))
1810 PO.(H.3200,0)
1815 W.(500)
1820 PO.(H.3E20,0)
1825 W.(5)
1830 B(0)=APE.(H.3E20)
1831 DI
1832 P.#3,F,#6,X,Y,C(1),C(2),C(3),B(0),W
1835 E=E+1
1840 C(4)=1
1845 Z(0)=0
2000 ;
2005 ;SEND DATA TO COORDINATOR
4000 T=3
4005 P=6
4010 M=P
4012 EI
4013 SC.(100,4055)
4015 IF M>0 G.4030
4020 M=P
4025 COS.5000
```

```
4030 IF Z(0)>0 G.4036
4035 G.4015
4036 DI
4037 IF Z(1)#32766 G.4040
4038 S.
4040 ;
4043 GOS.6000
4045 O=O+1
4048 C(4)=O
4050 G.520
4055 M=M-T
4060 RTI
4070 S.
4100 ;
4105 ;DATA TO COORDINATOR
5000 Y(0)=7
5005 Y(1)=O
5010 Y(2)=C(4)
5015 Y(3)=X
5020 Y(4)=C(1)
5025 Y(5)=C(2)
5030 Y(6)=C(3)
5035 Y(7)=B(0)
5050 GOS.30000
5055 R.
5900 ;
5910 ;DATA FROM COORDINATOR
6000 X=Z(1)
6005 Y=Z(2)
6010 R.
9000 ;
9005 ;I-MIC - LSI LINK ROUTINE
30000 IF Y(0)<=0 G.30020
30010 IF Y(0)<=127 G.30040
30020 P.'S/R@30000-INVALID WORD COUNT SUPPLIED AS',Y(0),'HENCE STOP'
30030 S.
30040 U=Y(0)*2
30060 GOS.32000
30070 F.J=1 TO Y(0)
30080 U=Y(J)
30090 GOS.32000
30100 U.H.2400(U)
30110 GOS.32000
30120 N.J
30125 U=0
30130 GOS.31000
30140 IF U=0 G.30170
30150 Z(0)=0
30160 R.
30170 U=0
30175 GOS.31000
30180 Z(0)=U/2
30190 IF U#2*Z(0) G.30210
30200 IF U<254 G.30230
30210 P.'S/R@30000-INVALID BYTE COUNT RECEIVED AS',U,'HENCE STOP'
30220 S.
30230 F.K=1 TO Z(0)
30235 U=0
```

```
30240 GOS.31000
30250 U.H.2400(U)
30260 GOS.31000
30270 U.H.2400(U)
30280 Z(K)=U
30290 N.K
30300 R.
31000 U.H.240A()
31010 U.H.C8(U)
31020 U.H.2412(U)
31030 U.H.240E()
31040 R.
32000 U.H.240A()
32010 U.H.C8(U)
32015 U.H.2412(K)
32020 U.H.240E()
32030 R.
```

```
 400 ;PROGRAM 071283
 405 P.'INTERACTION PREDICTION METHOD WITH GLOBAL FEEDBABCK'
 410 P.
 415 P.'LOCAL DECISION UNIT 2'
 420 P.
 425 INP.'DATE'A
 430 INP.'TIME'B
 435 P.
 440 P.
 445 P.'  N     V1     V2    C21    C22    C23    P2    Y21*'
 446 P.'************************************************'
 500 D.(3,2,4,4,5,3,15,21,21,21,7,1,10,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,20,10)
2000 ;
2005 ;INITIALISATION
2006 O=1 : E=1
2008 INP.'V1'X,'V2'Y
2010 F.I=1 TO 10
2012 M(I)=0 : N.I
2015 F.I=1 TO 7
2017 K(I)=0 : N.I
2020 F.I=1 TO 21
2022 H(I)=0,I(I)=0,J(I)=0 : N.I
2025 F.I=1 TO 5
2027 E(I)=0 : N.I
2028 C(4)=0
2029 Z(0)=0
2030 ;
2031 ;DECISION UNIT 2 OPTIMISATION ROUTINE
2033 ; DETERMINE C22 MAX(M(7));C22 MIN(M(8));C23 MAX(M(9));C23 MIN(M(10))
2035 M(1)=X-Y+1000
2040 D=3*(X+1000)
2042 D=MOD(D,6)
2043 IF D>=3 GOTO 2047
2044 IF D<=-3 G.2049
2045 D=0
2046 G.2051
2047 D=1
2048 GOTO 2051
2049 D=-1
2051 M(2)=(X+1000)/2+D
2055 M(3)=M(2)-1000
2060 M(4)=X-Y-1000
2070 M(5)=X-2*Y+2000
2080 M(6)=X-2*Y-2000
2090 IF M(1)<=M(2) G.2120
2100 M(7)=M(2)
2110 GOTO 2130
2120 M(7)=M(1)
2130 IF M(7)<=1000 G.2150
2140 M(7)=1000
2150 IF M(3)<=M(4) G.2180
2160 M(8)=M(3)
2170 GOTO 2200
2180 M(8)=M(4)
2200 IF M(5)<=1000 G.2230
2210 M(9)=1000
2220 G.2250
2230 M(9)=M(5)
```

```
2250 IF M(6)<=-1000 GOTO 2280
2260 M(10)=M(6)
2270 G.2298
2280 M(10)=-1000
2298 K(0)=-2*Y, K(1)=3*X-Y
2299 K(2)=3*(X-2*Y-5000)
2300 K(3)=3*(-X-5000)
2301 K(4)=MOD(K(1),6)
2302 K(5)=MOD(K(0),6)
2303 K(6)=MOD(K(2),6)
2304 K(7)=MOD(K(3),6)
2305 FOR I=4 TO 7
2306 IF K(I)>=3 GOTO 2310
2307 IF K(I)<=-3 G.2312
2308 K(I)=0
2309 G.2313
2310 K(I)=1
2311 G.2313
2312 K(I)=-1
2313 NEXT I
2314 H(1)=K(1)/6+K(4), I(1)=-Y/3+K(5)
2316 H(2)=X-Y-1000, I(2)=(X-2*Y-5000)/2+K(6)
2320 H(3)=X-Y+1000, I(3)=I(2)+2000
2330 H(4)=-1000, I(4)=(-X-5000)/2+K(7)
2340 H(5)=1000, I(5)=I(4)+2000
2350 H(6)=H(1)+667, I(6)=-1000
2360 H(7)=H(1)+1333, I(7)=1000
2370 H(8)=H(2), I(8)=X-2*Y-2000
2380 H(9)=H(3), I(9)=X-2*Y+2000
2390 H(10)=H(4), I(10)=-2000-X
2400 H(11)=H(5), I(11)=2000-X
2410 H(12)=-I(4)-3000, I(12)=I(6)
2420 H(13)=-I(5), I(13)=I(7)
2430 H(14)=H(8), I(14)=I(12)
2440 H(15)=H(14), I(15)=I(13)
2450 H(16)=H(9), I(16)=I(14)
2460 H(17)=H(16), I(17)=I(15)
2470 H(18)=H(10), I(18)=I(16)
2480 H(19)=H(18), I(19)=I(17)
2490 H(20)=H(11), I(20)=I(18)
2500 H(21)=1000, I(21)=1000
2510 F.I=1 TO 21
2520 IF H(I)>M(7) G.2590
2530 IF H(I)<M(8) G.2590
2540 IF I(I)>M(9) G.2590
2550 IF I(I)<M(10) GOTO 2590
2560 E(3)=(H(I)-X+Y)*(H(I)-X+Y)/1000
2565 D=E(3)+H(I)*H(I)/1000+I(I)*I(I)/1000
2570 E(4)=D-(1050*X/1000)*X/1000-2040
2580 IF E(4)<=0 G.2610
2590 J(I)=32767, H(I)=J(I), I(I)=H(I)
2600 G.2645
2610 E(1)=2*(Y-2000)*(Y-2000)/1000
2620 E(2)=(2*H(I)-I(I)-X-3000)*(2*H(I)-I(I)-X-3000)/1000
2630 J(I)=E(1)+E(2)+E(3)+H(I)*H(I)/1000+I(I)*I(I)/1000+2
2645 N.I
2650 J(0)=32766
2660 FOR I=1 TO 21
```

```
2670 IF J(I-1)>J(I) GOTO 2700
2680 J(I)=J(I-1), I(I)=I(I-1), H(I)=H(I-1)
2690 GOTO 2710
2700 J(I)=J(I), I(I)=I(I), H(I)=H(I)
2710 NEXT I
2720 E(5)=H(21)-X+Y
2723 G.2800
2725 P.
2730 PRINT C21 =',E(5),'   C22 =',H(21),'   C23 =',I(21)
2731 P.'P2*',J(21)
2732 P.
2735 ;
2740 ;NEW OPTIMISATION ROUTINE (2-FEB-84)
2745 H(21)=(2*X-Y+3000)/4
2750 I(21)=-(Y+3000)/4
2755 IF H(21)>M(7) G.2767
2760 IF H(21)<M(8) G.2768
2765 G.2770
2767 H(21)=M(7) : G.2770
2768 H(21)=M(8)
2770 IF I(21)>M(9) G.2776
2772 IF I(21)<M(10) G.2778
2774 G.2780
2776 I(21)=M(9) : G.2780
2778 I(21)=M(10)
2780 E(5)=H(21)-X+Y
2785 J(21)=E(5)*E(5)/1000+2*(Y-2000)*(Y-2000)/1000
2787 J(21)=J(21)+(2*H(21)-I(21)-X-3000)*(2*H(21)-I(21)-X-3000)/1000
2789 J(21)=J(21)+H(21)*H(21)/1000+I(21)*I(21)/1000+2
2790 ;
2795 ;SYNCHRONISATION ROUTINE
2800 EI
2805 GOS.5000
2810 IF Z(0)>0 G.2825
2815 W.(10)
2820 G.2805
2825 IF Z(1)#32766 G.2835
2830 S.
2835 Q=Q+1
2840 Z(0)=0
2845 W=(11*H(21)/10-13*E(5)/10+66*Z(2)/100-154*Z(1)/100)*100/98
2890 ;
2895 ;SEND CONTROLS TO TR48 AND TAKE REAL MEASUREMENTS
3000 APO.(H.3E08,E(5))
3005 APO.(H.3E0A,H(21))
3010 PO.(H.3200,0)
3015 W.(500)
3020 PO.(H.3E22,0)
3025 W.(5)
3030 B(0)=APE.(H.3E22)
3032 DI
3035 P.#3,E,#6,X,Y,E(5),H(21),I(21),J(21),B(0),W
3040 E=E+1
3045 C(4)=1
3050 Z(0)=0
3060 ;
3065 ;SEND DATA TO COORDINATOR
4000 T=3
```

```
4005 P=6
4010 M=P
4012 EI
4013 SC.(100,4055)
4015 IF M>0 G.4030
4020 M=P
4025 GOS.5000
4030 IF Z(0)>0 G.4036
4035 G.4015
4036 DI
4037 IF Z(1)#32766 G.4040
4038 S.
4040 ;
4043 GOS.6000
4045 Q=Q+1
4048 C(4)=0
4050 G.2029
4055 M=M-T
4060 RTI
4070 S.
4900 ;
4904 ;DATA TO COORDINATOR
5000 Y(0)=8
5005 Y(1)=Q
5010 Y(2)=C(4)
5015 Y(3)=Y
5020 Y(4)=E(5)
5025 Y(5)=H(21)
5030 Y(6)=I(21)
5035 Y(7)=J(21)
5040 Y(8)=B(0)
5050 GOS.30000
5055 R.
5900 ;
5910 ;DATA FROM COORDINATOR
6000 X=Z(1)
6005 Y=Z(2)
6010 R.
9000 ;
9005 ;I-MIC - LSI LINK ROUTINE
30000 IF Y(0)<=0 G.30020
30010 IF Y(0)<=127 G.30040
30020 P.'S/@30000-INVALID WORD COUNT SUPPLIED AS',Y(0),'HENCE STOP'
30030 S.
30040 U=Y(0)*2
30060 GOS.32000
30070 F.J=1 TO Y(0)
30080 U=Y(J)
30090 GOS.32000
30100 U.H.2400(U)
30110 GOS.32000
30120 N.J
30125 U=0
30130 GOS.31000
30140 IF U=0 G.30170
30150 Z(0)=0
30160 R.
30170 U=0
```

```
30175 GOS.31000
30180 Z(0)=U/2
30190 IF U#2*Z(0) G.30210
30200 IF U<254 G.30230
30210 P.'S/@30000-INVALID BYTE COUNT RECEIVED AS',U,'HENCE STOP'
30220 S.
30230 F.K=1 TO Z(0)
30235 U=0
30240 GOS.31000
30250 U.H.2400(U)
30260 GOS.31000
30270 U.H.2400(U)
30280 Z(K)=U
30290 N.K
30300 R.
31000 U.H.240A()
31010 U.H.C8(U)
31020 U.H.2412(U)
31030 U.H.240E()
31040 R.
32000 U.H.240A()
32010 U.H.C8(U)
32015 U.H.2412(K)
32020 U.H.240E()
32030 R.
```

C2. Local Decision Units (I-MICs) Software Listing

for

Interaction Prediction Method with Local Feedback

```
400 ; PROGRAM 190384
405 P.'INTERACTION PREDICTION METHOD WITH LOCAL FEEDBACK (SY)'
410 P.
415 P.'LOCAL DECISION UNIT 1'
420 P.
425 INP.'DATE'A
430 INP.'TIME',B
431 X=1171, Y=965, S=0
435 P.
440 P.
445 G.500
450 INP.'U11'Y,'Y11'X,'SHIFT'S
500 D.(3,2,4,4,5,3,15,21,21,21,7,1,10,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,20,10)
501 ;
506 ;INITIALISATION
510 E=0, C(0)=0, Q=1
520 F.I=1 TO 10
530 M(I)=0, Y(I)=0, Z(I)=0 : N.I
540 F.I=1 TO 3
550 A(I)=0, C(I)=0, E(I)=0, B(I-1)=0 : N.I
560 F.I=1 TO 5
570 H(I)=0, I(I)=0, J(I)=0 : N.I
580 INP.'K1'M(1),'K2'M(2)
600 P.#3,0,'  ITERATION :  U11=',Y,'   Y11=',X,'   SHIFT=',S
605 P.
610 P.
615 P.'  N    C11    C12    S1    P1    U11*    Y11*
800 Z(0)=0, C(4)=0
 805 E=E+1, C(0)=C(0)+1
 900 ;
1000 ; LOCAL DECISION UNIT 1 - OPTIMISATION ROUTINE
1005 H(1)=-2*Y/3+S/3+333, I(1)=-H(1)
1010 H(2)=-13*Y/10+S/2+900, I(2)=800-3*Y/5
1015 H(3)=1000, I(3)=-1000
1020 H(4)=-1000, I(4)=1000
1025 H(5)=-1000, I(5)=I(2)
1027 A(1)=X-1000
1030 F.I=1 TO 5
1035 IF ABS(H(I))>1000 G.1075
1040 IF ABS(I(I))>1000 G.1075
1045 A(3)=I(I)+3*Y/5
1050 IF A(3)>800 G.1075
1060 A(2)=H(I)*H(I)/1000+I(I)*I(I)/1000
1065 J(I)=A(1)*A(1)/1000+A(2)+1
1070 G.1080
1075 J(I)=32766
1080 N.I
1100 J(0)=32766
1105 F.I=1 TO 5
1110 IF J(I-1)>J(I) G.1120
1115 J(I)=J(I-1), H(I)=H(I-1), I(I)=I(I-1)
1120 N.I
1123 G.1700
1125 P.'C11=',H(5),'    C12=',I(5)
1130 P.'U11=',Y,'    Y11=',X
1135 P.'P1(X1000)=',J(5)
1700 ;
1705 ;SYNCHRONISATION ROUTINE
```

```
1075 J(I)=32766
1080 N. I
1100 J(0)=32766
1105 F. I=1 TO 5
1110 IF J(I-1)>J(I) G. 1120
1115 J(I)=J(I-1), H(I)=H(I-1), I(I)=I(I-1)
1120 N. I
1123 G. 1700
1125 P. 'C11=', H(5), '     C12=', I(5)
1130 P. 'U11=', Y, '     Y11=', X
1135 P. 'P1(X1000)=', J(5)
1700 ; MODEL & REAL PROCESS INTERFACE
1720 EI
1725 GOS. 5000
1730 IF Z(0)>0 G. 1745
1735 W. (20)
1740 G. 1725
1743 Z(0)=0
1745 IF Z(1)#32766 G. 1755
1747 P.
1748 P. 'TOTAL ITERATIONS REQUIRED :  ', C(0)
1750 S.
1755 Q=Q+1
1765 W=(6*I(5)/10-14*H(5)/10+99*Z(2)/50-117*Z(1)/50)*50/49
1799 ; SEND DATA TO ANALOGUE COMPUTER
1800 APO. (H. 3E04, H(5))
1805 APO. (H. 3E06, I(5))
1810 PO. (H. 3200, 0)
1815 W. (500)
1820 PO. (H. 3E20, 0) : W. (5)
1824 B(0)=APE. (H. 3E20)
1826 PO. (H. 3E22, 0) : W. (5)
1828 B(1)=APE. (H. 3E22)
1831 DI
1832 P. #3, E, #6, H(5), I(5), S, J(5), B(1), B(0), W
1840 E(1)=X-B(0)
1850 IF ABS(E(1))>3 G. 2000
1860 P. 'AFTER  ', #3, E, '  ITERATIONS'
1870 P. H(5), I(5), S, J(5), B(1), B(0), W
1880 C(4)=1,  Z(0)=0
1890 G. 3900
2000 ; UPDATING ROUTINE
2010 E(2)=Y-B(1)
2020 C(1)=S+M(1)*E(1)/1000+M(2)*E(2)/1000
2030 S=C(1)
2040 G. 800
3900 ; SEND CONVERGED SOLUTION TO COORDINATOR
4000 T=3
4005 P=6
4010 M=P
4012 EI
4013 SC. (100, 4055)
4015 IF M>0 G. 4030
4020 M=P
4025 GOS. 5000
4030 IF Z(0)>0 G. 4036
4035 G. 4015
4036 DI
4037 IF Z(1)#32766 G. 4040
4038 P.
```

```
4039 P. 'TOTAL ITERATIONS REQUIRED :   ',C(0)
4040 ;
4043 GOS. 6000
4045 Q=Q+1
4048 C(4)=0
4050 G. 605
4055 M=M-T
4060 RTI
4070 S.
5000 Y(0)=8
5005 Y(1)=Q
5010 Y(2)=C(4)
5015 Y(3)=X
5020 Y(4)=H(5)
5025 Y(5)=I(5)
5030 Y(6)=J(5)
5035 Y(7)=B(0)
5040 Y(8)=S
5050 GOS. 30000
5055 R.
6000 X=Z(1)
6005 Y=Z(2)
6010 R.
30000 IF Y(0)<=0 G. 30020
30010 IF Y(0)<=127 G. 30040
30020 P. 'S/R@30000-INVALID WORD COUNT SUPPLIED AS', Y(0), 'HENCE STOP'
30030 S.
30040 U=Y(0)*2
30060 GOS. 32000
30070 F. J=1 TO Y(0)
30080 U=Y(J)
30090 GOS. 32000
30100 U. H. 2400(U)
30110 GOS. 32000
30120 N. J
30125 U=0
30130 GOS. 31000
30140 IF U=0 G. 30170
30150 Z(0)=0
30160 R.
30170 U=0
30175 GOS. 31000
30180 Z(0)=U/2
30190 IF U#2*Z(0) G. 30210
30200 IF U<254 G. 30230
30210 P. 'S/R@30000-INVALID BYTE COUNT RECEIVED AS', U, 'HENCE STOP'
30220 S.
30230 F. K=1 TO Z(0)
30235 U=0
30240 GOS. 31000
30250 U. H. 2400(U)
30260 GOS. 31000
30270 U. H. 2400(U)
30280 Z(K)=U
30290 N. K
30300 R.
31000 U. H. 240A()
31010 U. H. C8(U)
31020 U. H. 2412(U)
31030 U. H. 240E()
```

```
31040 R.
32000 U. H. 240A( )
32010 U. H. C8(U)
32015 U. H. 2412(K)
32020 U. H. 240E( )
32030 R.
```

```
400 ;PROGRAM 190384
405 P.'INTERACTION PREDICTION METHOD WITH LOCAL FEEDBACK (SY)'
410 P.
415 P.'LOCAL DECISION UNIT 2'
420 P.
425 INP.'DATE'A
430 INP.'TIME'B
432 X=1171, Y=965, S=0
435 P.
440 P.
445 G.500
450 INP.'U21'X,'Y21'Y,'SHIFT'S
500 D.(3,2,4,4,5,3,15,21,21,21,21,21,10,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,20,10
501 ;
505 ;INITIALISATION
510 E=0, C(0)=0, Q=1
530 F.I=1 TO 18
540 H(I)=0, I(I)=0, J(I)=0, K(I)=0, L(I)=0 : N.I
550 F.I=1 TO 10
560 M(I)=0, Y(I)=0, Z(I)=0 : N.I
570 F.I=1 TO 4
580 C(I)=0, F(I)=0 : N.I
590 A(1)=0,A(2)=0,A(3)=0
595 INP.'K1'M(1),'K2'M(2)
600 P.#3,Q,'  ITERATION :  U21=',X,'   Y21=',Y,'   SHIFT=',S
605 P.
610 P.
615 P.'  N    C21    C22    C23    Y22    S2    P2    U21*    Y21*
 800 Z(0)=0, C(4)=0
 805 E=E+1, C(0)=C(0)+1
 900 ;
1000 ;LOCAL DECISION UNIT 2 - OPTIMISATION ROUTINE
1005 H(1)=-4*X/11+6*S/11+1636, I(1)=5*X/11-2*S/11+455
1010 J(1)=-X/22-2*X/11-1046, K(1)=-X/22-2*S/11+1955
1015 H(2)=1000, I(2)=3*X/5-2*S/5+200
1020 J(2)=X/10-2*S/5+1300, K(2)=X/10-2*S/5+1700
1025 H(3)=-2*X/3+2*S/3+2000, I(3)=1000
1030 J(3)=-X/2-500, K(3)=-X/2+2500
1035 H(4)=-1000, I(4)=3*X/5-2*S/5-600
1040 J(4)=X/10-2*S/5-2100, K(4)=X/10-2*S/5+900
1045 H(5)=-2*X/3+2*S/3+667, I(5)=-1000
1050 J(5)=-X/2-2500, K(5)=-X/2+500
1055 H(6)=1000, I(6)=-1000, J(6)=J(5), K(6)=K(5)
1060 H(7)=1000, I(7)=4*X/7-2*S/7+286
1065 J(7)=-1000, K(7)=X/7-4*S/7+1571
1070 H(8)=H(3), I(8)=1000, J(8)=-1000, K(8)=3000-X
1075 H(9)=-1000, I(9)=1000, J(9)=J(3), K(9)=K(3)
1080 H(10)=-1000, I(10)=I(7), J(10)=1000, K(10)=X/7-4*S/7-429
1085 H(11)=H(5), I(11)=-1000, J(11)=1000, K(11)=-3000-X
1090 H(12)=1000, I(12)=-1000, J(12)=-1000, K(12)=-X-1000
1095 H(13)=1000, I(13)=-K(5), J(13)=-1000, K(13)=0
1100 H(14)=1000, I(14)=1000, J(14)=-1000, K(14)=K(8)
1105 H(15)=-1000, I(15)=1000, J(15)=-1000, K(15)=K(8)
1110 H(16)=-1000, I(16)=1000, J(16)=1000, K(16)=-X+1000
1115 H(17)=-1000, I(17)=-1000, J(17)=1000, K(17)=K(11)
1120 H(18)=1000, I(18)=-1000, J(18)=1000, K(18)=K(11)
1125 A(1)=Y-2000
1200 F.I=1 TO 18
```

```
1205 IF ABS(H(I))>1000 G.1245
1210 IF ABS(I(I))>1000 G.1245
1215 IF ABS(J(I))>1000 G.1245
1220 IF K(I)<0 G.1245
1225 A(2)=K(I)-3000, A(2)=A(2)*A(2)/1000
1230 A(3)=H(I)*H(I)/1000+I(I)*I(I)/1000+J(I)*J(I)/1000
1235 L(I)=2*(A(1)*A(1)/1000)+A(2)+A(3)+2
1240 G.1250
1245 L(I)=32766
1250 N.I
1260 L(0)=32766
1265 F.I=1 TO 18
1270 IF L(I-1)>L(I) G.1280
1275 L(I)=L(I-1), H(I)=H(I-1), I(I)=I(I-1), J(I)=J(I-1), K(I)=K(I-1)
1280 N.I
1283 G.2800
1285 P.'C21=',H(18),'    C22=',I(18),'    C23=',J(18)
1290 P.'U21=',X,'    Y21=',Y,'    Y22=',K(18)
1295 P.'P2(X1000)=',L(18)
2800 ;
2801 ;SYNCHRONISATION ROUTINE
2802 EI
2805 COS.5000
2810 IF Z(0)>0 G.2825
2815 W.(20)
2820 G.2805
2825 IF Z(1)#32766 G.2835
2827 P.
2828 P.'TOTAL ITERATIONS REQUIRED :  ',C(0)
2830 S.
2835 Q=Q+1
2840 Z(0)=0
2845 W=(11*I(18)/10-13*H(18)/10+66*Z(2)/100-154*Z(1)/100)*100/98
2952 ;
2953 ;SEND CONTROLS TO TR48 AND TAKE REAL MEASUREMENTS
3000 APO.(H.3E08,H(18))
3005 APO.(H.3E0A,I(18))
3010 PO.(H.3200,0)
3015 W.(500)
3020 PO.(H.3E20,0) : W.(5)
3025 B(0)=APE.(H.3E20)
3027 PO.(H.3E22,0) : W.(5)
3030 B(1)=APE.(H.3E22)
3032 DI
3035 P.#3,E,#6,H(18),I(18),J(18),K(18),S,L(18),B(0),B(1),W
3037 ;
3038 ;CHECK CONVERGENCE
3040 E(1)=Y-B(1)
3050 IF ABS(E(1))>3 G.3500
3060 P.'AFTER  ',#3,E,'  ITERATIONS'
3070 P.H(18),I(18),J(18),K(18),S,L(18),B(0),B(1),W
3080 C(4)=1, Z(0)=0
3090 G.3900
3490 ;
3500 ;UPDATING ROUTINE
3510 E(2)=X-B(0)
3520 C(1)=S+M(1)*E(1)/1000+M(2)*E(2)/1000
3530 S=C(1)
```

```
3540 G.800
3890 ;
3900 ;SEND CONVERGED SOLUTION TO COORDINATOR
4000 T=3
4005 P=6
4010 M=P
4012 EI
4013 SC.(100,4055)
4015 IF M>0 G.4030
4020 M=P
4025 GOS.5000
4030 IF Z(0)>0 G.4036
4035 G.4015
4036 DI
4037 IF Z(1)#32766 G.4043
4038 P.
4039 P.'TOTAL ITERATIONS REQUIRED :  ',C(0)
4040 S.
4043 GOS.6000
4045 Q=Q+1
4050 G.605
4055 M=M-T
4060 RTI
4070 S.
4900 ;
4905 ;DATA TO COORDINATOR
5000 Y(0)=10
5005 Y(1)=Q
5010 Y(2)=C(4)
5015 Y(3)=Y
5020 Y(4)=H(18)
5025 Y(5)=I(18)
5030 Y(6)=J(18)
5035 Y(7)=K(18)
5040 Y(8)=L(18)
5045 Y(9)=B(1)
5047 Y(10)=S
5050 GOS.30000
5055 R.
5900 ;
5910 ;DATA FROM COORDINATOR
6000 X=Z(1)
6005 Y=Z(2)
6010 R.
9000 ;
9005 ;I-MIC - LSI LINK ROUTINE
30000 IF Y(0)<=0 G.30020
30010 IF Y(0)<=127 G.30040
30020 P.'S/R@30000-INVALID WORD COUNT SUPPLIED AS',Y(0),'HENCE STOP'
30030 S.
30040 U=Y(0)*2
30060 GOS.32000
30070 F.J=1 TO Y(0)
30080 U=Y(J)
30090 GOS.32000
30100 U.H.2400(U)
30110 GOS.32000
30120 N.J
```

```
30125 U=0
30130 GOS.31000
30140 IF U=0 G.30170
30150 Z(0)=0
30160 R.
30170 U=0
30175 GOS.31000
30180 Z(0)=U/2
30190 IF U#2*Z(0) G.30210
30200 IF U<254 G.30230
30210 P.'S/R@30000-INVALID BYTE COUNT RECEIVED AS',U,'HENCE STOP'
30220 S.
30230 F.K=1 TO Z(0)
30235 U=0
30240 GOS.31000
30250 U.H.2400(U)
30260 GOS.31000
30270 U.H.2400(U)
30280 Z(K)=U
30290 N.K
30300 R.
31000 U.H.240A()
31010 U.H.C8(U)
31020 U.H.2412(U)
31030 U.H.240E()
31040 R.
32000 U.H.240A()
32010 U.H.C8(U)
32015 U.H.2412(K)
32020 U.H.240E()
32030 R.
```

C3. Local Decision Units (I-MICs) Software Listing

for

Interaction Balance Method with Global Feedback

```
1000 ;PROGRAM 160284
1005 P.'INTERACTION BALANCE METHOD WITH GLOBAL FEEDBACK'
1010 P.
1015 P.'LOCAL DECISION UNIT 1'
1020 P.
1025 INP.'DATE'A
1030 INP.'TIME'B
1035 X=-2538,Y=-740
1036 G.1043
1040 P.
1042 INP.'L1'X,'L2'Y
1043 P.
1044 P.
1045 P.'  N    L1     L2    C11     C12    P1    Y11*'
1050 P.'**************************************************************'
4002 D.(4,-1,-1,-1,-1,-1,-1,11,11,11,11,11,20,20,20,20,20,20,20,-1,-1,-1,-1,-1,11
4010 ;
4011 ;INITIALISATION
4012 O=1
4015 E=1
4020 A(0)=0, A(1)=0, A(2)=0, A(3)=0, A(4)=0
4035 F.I=1 TO 11
4040 H(I)=0, I(I)=0, J(I)=0, K(I)=0, L(I)=0
4045 N.I
4059 M(1)=0
4070 Z(0)=0
4998 ;
4999 ;DECISION UNIT 1 OPTIMISATION ROUTINE
5000 H(1)=X/4, I(1)=-H(1)
5002 J(1)=500-375*X/1000+Y/4
5003 K(1)=1000-X/4+Y/2
5005 H(2)=1000, I(2)=-1000
5007 J(2)=-500-X/8+Y/4
5008 K(2)=1000-X/4+Y/2
5010 H(3)=-80+520*X/2992+72*Y/2992
5011 I(3)=347+240*X/2992-312*Y/2992
5012 J(3)=754-400*X/2992+520*Y/2992
5013 K(3)=1080-520*X/2992+1424*Y/2992
5015 H(4)=-1000, I(4)=146+60*X/1424-156*Y/1424
5017 J(4)=1089-100*X/1424+260*Y/1424
5018 K(4)=1033-260*X/1424+676*Y/1424
5020 H(5)=-3000+Y/4, I(5)=-1000
5022 J(5)=3000, K(5)=4000+Y/4
5025 H(6)=1000, I(6)=-1000
5027 J(6)=3000, K(6)=8000
5030 H(7)=-1000, I(7)=-1000
5032 J(7)=3000, K(7)=6000
5035 H(8)=-134+260*X/1424
5036 I(8)=584+60*X/1424
5037 J(8)=359-100*X/1424
5038 K(8)=0
5040 H(9)=-1000, I(9)=385
5042 J(9)=692, K(9)=0
5045 H(10)=X/4, I(10)=-H(10)
5047 J(10)=I(10), K(10)=0
5050 H(11)=1000, I(11)=-1000
5052 J(11)=-1000, K(11)=0
5070 F.I=1 TO 11
```

```
5075 IF ABS(H(I))>1000 G.5125
5077 IF ABS(I(I))>1000 G.5125
5079 A(4)=I(I)+6*J(I)/10
5080 IF A(4)>800 G.5125
5085 IF K(I)<0 G.5125
5105 A(1)=(K(I)-1000)*(K(I)-1000)/10000
5110 A(2)=H(I)*H(I)/10000+I(I)*I(I)/10000
5115 A(3)=X*J(I)/10000-Y*K(I)/10000
5120 L(I)=A(1)+A(2)+A(3)+2
5121 G.5135
5125 L(I)=32766
5135 N.I
5200 L(0)=32766
5205 F.I=1 TO 11
5210 IF L(I-1)>L(I) G.5240
5215 L(I)=L(I-1), H(I)=H(I-1), I(I)=I(I-1)
5220 J(I)=J(I-1), K(I)=K(I-1)
5240 NEXT I
5300 A(1)=(K(11)-1000)*(K(11)-1000)/1000
5305 A(2)=H(11)*H(11)/1000+I(11)*I(11)/1000
5310 A(3)=X*J(11)/1000-Y*K(11)/1000
5315 L(11)=A(1)+A(2)+A(3)+2
5320 G.7000
6500 P.'C11=',H(11),'    C12=',I(11)
6505 P.'U11=',J(11),'    Y11=',K(11)
6510 P.'P1=',L(11)
6520 S.
6900 ;
6905 ;SYNCHRONISATION ROUTINE
7000 EI
7005 W.(100)
7010 GOS.8500
7015 IF Z(0)>0 G.7030
7020 W.(10)
7025 G.7010
7030 IF Z(1)#32766 G.7040
7035 S.
7040 Q=Q+1
7045 Z(0)=0
7050 W=(33*I(11)/50-77*H(11)/50-13*Z(1)/10+11*Z(2)/10)*50/49
7052 ;
7053 ;SEND CONTROLS TO TR48 AND TAKE REAL MEASUREMENTS
7055 APO.(H.3E04,H(11))
7060 APO.(H.3E06,I(11))
7065 PO.(H.3200,0)
7070 W.(500)
7075 PO.(H.3E20,0)
7080 W.(5)
7085 M(0)=APE.(H.3E20)
7090 DI
7095 P.#3,E,#6,X,Y,H(11),I(11),L(11),M(0),W,'
7100 E=E+1
7105 M(1)=1
7110 Z(0)=0
8000 ;
8005 ;SEND DATA TO COORDINATOR
8010 T=3
8015 P=6
```

```
8020 M=P
8025 EI
8030 SC.(100,8080)
8035 IF M>0 G.8050
8040 M=P
8045 GOS.8500
8050 IF Z(0)>0 G.8056
8055 G.8035
8056 DI
8057 IF Z(1)#32766 G.8060
8058 S.
8060 ;
8065 GOS.8700
8070 O=O+1
8071 Z(0)=0
8072 M(1)=0
8075 G.4999
8080 M=M-T
8085 RTI
8090 S.
8400 ;
8405 ;DATA TO COORDINATOR
8500 Y(0)=8
8505 Y(1)=O
8510 Y(2)=M(1)
8515 Y(3)=X
8520 Y(4)=J(11)
8525 Y(5)=H(11)
8530 Y(6)=I(11)
8535 Y(7)=K(11)
8540 Y(8)=M(0)
8560 GOS.30000
8570 R.
8600 ;
8605 ;DATA FROM COORDINATOR
8700 X=Z(1)
8705 Y=Z(2)
8710 R.
9000 ;
9005 ;I-MIC - LSI LINK ROUTINE
30000 IF Y(0)<=0 G.30020
30010 IF Y(0)<=127 G.30040
30020 P.'S/R@30000-INVALID WORD COUNT SUPPLIED AS',Y(0),'HENCE STOP'
30030 S.
30040 U=Y(0)*2
30060 GOS.32000
30070 F.J=1 TO Y(0)
30080 U=Y(J)
30090 GOS.32000
30100 U.H.2400(U)
30110 GOS.32000
30120 N.J
30125 U=0
30130 GOS.31000
30140 IF U=0 G.30170
30150 Z(0)=0
30160 R.
30170 U=0
```

```
30175 GOS.31000
30180 Z(0)=U/2
30190 IF U#2*Z(0) G.30210
30200 IF U<254 G.30230
30210 P.'S/R@30000-INVALID BYTE COUNT RECEIVED AS'U,'HENCE STOP'
30220 S.
30230 F.K=1 TO Z(0)
30235 U=0
30240 GOS.31000
30250 U.H.2400(U)
30260 GOS.31000
30270 U.H.2400(U)
30280 Z(K)=U
30290 N.K
30300 R.
31000 U.H.240A()
31010 U.H.C8(U)
31020 U.H.2412(U)
31030 U.H.240E()
31040 R.
32000 U.H.240A()
32010 U.H.C8(U)
32015 U.H.2412(K)
32020 U.H.240E()
32030 R.
```

```
1000 ;PROGRAM 160284
1005 P.'INTERACTION BALANCE METHOD WITH GLOBAL FEEDBACK'
1010 P.
1015 P.'LOCAL DECISION UNIT 2'
1020 P.
1025 INP.'DATE'A
1030 INP.'TIME'B
1035 X=-2538,Y=-740
1036 G.1043
1040 P.
1042 INP.'L1'X,'L2'Y
1043 P.
1044 P.
1045 P.'  N    L1     L2     C21    C22  C23     P2    Y21*'
1050 P.'**************************************************************'
4002 D.(4,-1,-1,-1,-1,-1,-1,11,11,11,11,11,20,20,20,20,20,20,20,-1,-1,-1,-1,-1,11
4010 ;
4011 ;INITIALISATION
4012 H(1)=0
4015 Q=1, E=1
4020 A(0)=0,A(1)=0,A(2)=0,A(3)=0,A(4)=0
4035 F.I=1 TO 20
4040 M(I)=0, N(I)=0, O(I)=0, P(I)=0, Q(I)=0, R(I)=0, S(I)=0
4045 N.I
4070 Z(0)=0
5000 ;
5005 ;DECISION UNIT 2 OPTIMISATION ROUTINE
6100 M(1)=1111+X/18+8*Y/18, N(1)=M(1)-Y
6101 O(1)=-1111+(Y-X)/18
6102 P(1)=1444+4*X/18-22*Y/18
6103 Q(1)=P(1)+Y, R(1)=1888+(Y-X)/18
6105 M(2)=1142+X/14+4*Y/14
6106 N(2)=1000, O(2)=-1142-X/14+3*Y/14
6107 P(2)=1285+X/7-6*Y/14
6108 Q(2)=1428+3*X/14-Y/7
6109 R(2)=1857-X/14+3*Y/14
6110 M(3)=750+Y/2, N(3)=750-Y/2
6112 O(3)=-750, P(3)=-Y
6113 Q(3)=0, R(3)=2250
6115 M(4)=1000, N(4)=1142+X/14-6*Y/14
6117 O(4)=-1142-(X+Y)/14, P(4)=1571+4*X/14-10*Y/14
6118 Q(4)=1428+3*X/14-4*Y/14, R(4)=1857-(X+Y)/14
6120 M(5)=666+Y/3, N(5)=1000
6122 O(5)=-666+Y/6, P(5)=333-Y/3
6123 Q(5)=0, R(5)=2333+Y/6
6125 M(6)=1200+X/10+Y/5, N(6)=1000
6127 O(6)=-1000, P(6)=1200+X/10-3*Y/10
6128 Q(6)=1400+X/5-Y/10, R(6)=1800-X/10+3*Y/10
6130 M(7)=1000, N(7)=667-Y/3
6132 O(7)=-666-Y/6, P(7)=N(7)-1000
6133 Q(7)=0, R(7)=2333-Y/6
6135 M(8)=1000, N(8)=1200+X/10-4*Y/10
6137 O(8)=-1000, P(8)=1600-7*Y/10+3*X/10
6138 Q(8)=1400-3*Y/10+X/5, R(8)=1800-(X+Y)/10
6140 M(9)=500+Y/4, N(9)=1000
6142 O(9)=-1000, P(9)=500-Y/4
6143 Q(9)=0, R(9)=2500+Y/4
6145 M(10)=-1000, N(10)=1000
```

```
6147 O(10)=-1000, P(10)=2666+(X-Y)/6
6148 Q(10)=P(10)-2000, R(10)=2000/6+(Y-X)/6
6150 M(11)=-1000, N(11)=1000
6152 O(11)=-1000, P(11)=-Y/2
6153 Q(11)=0, R(11)=3000+Y/2
6155 M(12)=X/6, N(12)=1000
6157 O(12)=-1000, P(12)=3000
6158 Q(12)=2000+M(12), R(12)=0
6160 M(13)=-1000, N(13)=1000
6162 O(13)=-1000, P(13)=4000+(X-Y)/4
6163 Q(13)=P(13)-2000, R(13)=0
6165 M(14)=1000, N(14)=1000
6167 O(14)=-1000, P(14)=1333+(X-Y)/6
6168 Q(14)=P(14), R(14)=1666+(Y-X)/6
6170 M(15)=1000, N(15)=X/6-Y/3
6172 O(15)=-1000, P(15)=1000+X/3-4*Y/6
6173 Q(15)=2000+N(15), R(15)=0
6175 M(16)=1000, N(16)=1000
6177 O(16)=-1000, P(16)=3000
6178 Q(16)=2000+(X-Y)/4, R(16)=0
6180 M(17)=1000, N(17)=-1000
6182 O(17)=-1000, P(17)=-1333+(X-Y)/6
6183 O(17)=P(17)+2000, R(17)=333+(Y-X)/6
6185 M(18)=1000, N(18)=-1000
6187 O(18)=-1000, P(18)=(X-Y)/4
6188 O(18)=2000+P(18), R(18)=0
6190 M(19)=1000, N(19)=500-Y/4
6192 O(19)=-1000, P(19)=-500-Y/4
6193 O(19)=0, R(19)=10000
6195 M(20)=1000, N(20)=-1000
6197 O(20)=-1000, P(20)=-1000
6198 O(20)=0, R(20)=3000+Y/2
6250 F.I=1 TO 20
6260 IF ABS(M(I))>1000 G.6323
6270 IF ABS(N(I))>1000 G.6323
6280 IF ABS(O(I))>1000 G.6323
6285 IF Q(I)<0 G.6323
6290 IF R(I)<0 G.6323
6305 A(1)=((Q(I)-2000)*(Q(I)-2000)/10000)*2
6310 A(2)=(R(I)-3000)*(R(I)-3000)/10000
6315 A(3)=M(I)*M(I)/10000+N(I)*N(I)/10000+O(I)*O(I)/10000
6320 A(4)=Y*P(I)/10000-X*Q(I)/10000
6321 A(0)=A(3)-(105*P(I)/1000)*P(I)/1000-204
6322 IF A(0)<=0 G.6325
6323 S(I)=32766
6324 G.6330
6325 S(I)=A(1)+A(2)+A(3)+A(4)+2
6330 N.I
6400 S(0)=32766
6405 F.I=1 TO 20
6410 IF S(I-1)>S(I) G.6440
6415 S(I)=S(I-1), M(I)=M(I-1), N(I)=N(I-1), O(I)=O(I-1)
6420 P(I)=P(I-1), Q(I)=Q(I-1), R(I)=R(I-1)
6440 NEXT I
6450 A(1)=((O(20)-2000)*(Q(20)-2000)/1000)*2
6455 A(2)=(R(20)-3000)*(R(20)-3000)/1000
6460 A(3)=M(20)*M(20)/1000+N(20)*N(20)/1000+O(20)*O(20)/1000
6465 A(4)=Y*P(20)/1000-X*Q(20)/1000
```

```
6470 S(20)=A(1)+A(2)+A(3)+A(4)+2
6500 G.7000
6505 P.'C21=',M(20),'    C22=',N(20),'    C23=',O(20)
6510 P.'U21',P(20),'    Y21=',Q(20),'    Y22=',R(20)
6515 P.'P2=',S(20)
6520 S.
6570 P.
6900 ;
6905 ;SYNCHRONISATION ROUTINE
7000 EI
7010 GOS.8500
7015 IF Z(0)>0 G.7030
7020 W.(10)
7025 G.7010
7030 IF Z(1)#32766 G.7040
7035 S.
7040 Q=Q+1
7045 Z(0)=0
7047 ;
7048 ;SEND CONTROLS TO TR48 AND TAKE REAL MEASUREMENTS
7050 W=(99*N(20)/50-117*M(20)/50-7*Z(1)/5+3*Z(2)/5)*50/49
7055 APO.(H.3E08,M(20))
7060 APO.(H.3E0A,N(20))
7065 PO.(H.3200,0)
7070 W.(500)
7075 PO.(H.3E22,0)
7080 W.(5)
7085 H(0)=APE.(H.3E22)
7090 DI
7095 P.#3,E,#6,X,Y,M(20),N(20),O(20),S(20),H(0),W
7100 E=E+1
7105 H(1)=1
7110 Z(0)=0
8000 ;
8005 ;SEND DATA TO COORDINATOR
8010 T=3
8015 P=6
8020 M=P
8025 EI
8030 SC.(300,8080)
8035 IF M>0 G.8050
8040 M=P
8045 GOS.8500
8050 IF Z(0)>0 G.8056
8055 G.8035
8056 DI
8057 IF Z(1)#32766 G.8060
8058 S.
8060 ;
8065 GOS.8700
8070 Q=Q+1, Z(0)=0
8072 H(1)=1
8075 G.5005
8080 M=M-T
8085 RTI
8090 S.
8400 ;
8405 ;DATA TO COORDINATOR
```

```
8500 Y(0)=10
8505 Y(1)=Q
8510 Y(2)=H(1)
8515 Y(3)=Y
8520 Y(4)=P(20)
8525 Y(5)=M(20)
8530 Y(6)=N(20)
8535 Y(7)=O(20)
8540 Y(8)=Q(20)
8545 Y(9)=R(20)
8550 Y(10)=H(0)
8560 GOS.30000
8570 R.
8600 ;
8605 ;DATA FROM COORDINATOR
8700 X=Z(1)
8705 Y=Z(2)
8710 R.
9000 ;
9005 ;I-MIC - LSI LINK ROUTINE
30000 IF Y(0)<=0 G.30020
30010 IF Y(0)<=127 G.30040
30020 P.'S/R@30000-INVALID WORD COUNT SUPPLIED AS',Y(0),'HENCE STOP'
30030 S.
30040 U=Y(0)*2
30060 GOS.32000
30070 F.J=1 TO Y(0)
30080 U=Y(J)
30090 GOS.32000
30100 U.H.2400(U)
30110 GOS.32000
30120 N.J
30125 U=0
30130 GOS.31000
30140 IF U=0 G.30170
30150 Z(0)=0
30160 R.
30170 U=0
30175 GOS.31000
30180 Z(0)=U/2
30190 IF U#2*Z(0) G.30210
30200 IF U<254 G.30230
30210 P.'S/R@30000-INVALID BYTE COUNT RECEIVED AS',U,'HENCE STOP'
30220 S.
30230 F.K=1 TO Z(0)
30235 U=0
30240 GOS.31000
30250 U.H.2400(U)
30260 GOS.31000
30270 U.H.2400(U)
30280 Z(K)=U
30290 N.K
30300 R.
31000 U.H.240A()
31010 U.H.C8(U)
31020 U.H.2412(U)
31030 U.H.240E()
31040 R.
```

```
32000 U.H.240A()
32010 U.H.C8(U)
32015 U.H.2412(K)
32020 U.H.240E()
32030 R.
```

C4. Local Decision Units (I-MICs) Software Listing

for

Interaction Balance Method with Local Feedback

```
  50 G.4000
  99 ;TR49 SETTING UP - ATTENUATORS ADJUSTMENT
 100 EI
 101 P=-610,Q=149,R=924,S=1000
 104 G.106
 105 INP.'C11'P,'C12'Q,'C21'R,'C22'S
 106 F.I=1 TO 10
 110 APO.(H.3F04,P)
 120 APO.(H.3F06,Q)
 160 PO.(H.3200,0)
 165 PO.(H.3F20,0):WAIT(5)
 170 A=APF.(H.3F20)
 175 PO.(H.3F22,0):WAIT(5)
 180 B=APF.(H.3F22)
 185 P.A,B
 200 WAIT(200)
 250 N.I
 270 G.106
 300 S.
4000 ;PROGRAM 30629
4001 ;INTERACTION BALANCE METHOD WITH LOCAL FEEDBACK
4002 ;LOCAL DECISION UNIT 1
4003 ;
4004 ;
4005 D.(4,4,10,5,5,-1,-1,7,7,7,-1,7,14,14,14,-1,14,14,14,-1,-1,-1,-1,-1,15,5)
4007 V=965,X=-2546,Y=-744
4008 G.4012
4010 INP.'U11'V,'LAMDA1'X,'LAMDA2'Y
4011 ;INITIALISATION
4012 Q=1
4015 F=0
4017 N=20
4020 F.I=0 TO 4
4022 A(I)=0,B(I)=0
4023 N.I
4025 F.I=1 TO 5
4027 M(I)=0,H(I)=0,Z(I)=0
4028 N.I
4030 F.I=1 TO 10
4032 C(I)=0
4033 N.I
4035 F.I=1 TO 7
4040 H(I)=0, I(I)=0, J(I)=0, L(I)=0.
4045 N.I
4058 P.
4059 P.
4060 P.'INTERACTION BALANCE METHOD WITH LOCAL FEEDBACK'
4062 P.
4064 P.'LOCAL DECISION UNIT 1'
4066 P.
4067 P.'DATE: 25-JUL-83'
4068 INP.'TIME'B
4069 P.
4070 P.'U11=965, LAMDA1=-2546, LAMDA2=-744'
4071 P.
4072 P.
4073 N(1)=32766
4075 P.'  N   LAMDA1   U11   U11*'
4077 ;GAIN COEFFICIENTS FOR CONSTANT GAIN
4078 C(0)=300
4079 C(6)=100
```

```
4080 Z(0)=0
4082 F=F+1
4999 ;LOCAL DECISION UNIT '1'
5000 H(1)=333+Y/6-2*V/3, I(1)=-H(1)
5002 J(1)=667+Y/3+2*V/3
5005 H(2)=1000, I(2)=-1000
5007 J(2)=2000+2*V
5010 H(3)=Y/4+900-13*V/10
5011 I(3)=900-6*V/10
5012 J(3)=Y/4+100+13*V/10
5015 H(4)=-V, I(4)=V, J(4)=0
5020 H(5)=1000, I(5)=-1000, J(5)=0
5025 H(6)=900-26*V/10, I(6)=I(3), J(6)=0
5030 H(7)=-1000, I(7)=I(3), J(7)=-1300+26*V/10
5070 F.I=1 TO 7
5075 IF ABS(H(I))>1000 G.5125
5077 IF ABS(I(I))>1000 G.5125
5079 A(4)=I(I)+6*V/10
5080 IF A(4)>900 G.5125
5085 IF J(I)<0 G.5125
5105 A(1)=(J(I)-1000)*(J(I)-1000)/10000
5110 A(2)=H(I)*H(I)/10000+I(I)*I(I)/10000
5115 A(3)=X*V/10000-Y*J(I)/10000
5120 L(I)=A(1)+A(2)+A(3)+2
5121 G.5135
5125 L(I)=32766
5135 N.I
5200 L(0)=32766
5205 F.I=1 TO 7
5210 IF L(I-1)>L(I) G.5240
5215 L(I)=L(I-1), H(I)=H(I-1)
5220 I(I)=I(I-1), J(I)=J(I-1)
5240 N.I
5250 A(1)=(J(7)-1000)*(J(7)-1000)/1000
5251 A(2)=H(7)*H(7)/1000+I(7)*I(7)/1000
5252 A(3)=Y*V/1000-Y*J(7)/1000
5253 L(7)=A(1)+A(2)+A(3)+2
5300 G.6602
5990 P.'C11=',H(7),'     C12=',I(7)
5991 P.'U11=',V,'     Y11=',J(7)
5992 P.'L1(X1000)=',L(7)
5993 P.
6600 ;MODEL & REAL PROCESS INTERFACE
6602 FI
6605 OUT.8500
6607 WAIT(10)
6610 IF Z(0)>0 G.6616
6615 G.6605
6616 IF Z(1)#32766 G.6620
6618 S.
6620 X=Z(1)
6622 W=Z(3)
6623 M(0)=Z(1), N(0)=Z(2)
6625 C=C+1
6750 APO.(H.3F04,H(7))
6755 APO.(H.3F06,I(7))
6767 PO.(H.3200,0)
6768 WAIT(500)
6769 PO.(H.3F20,0)
6770 WAIT(5)
6775 R(1)=APF.(H.3F20)
6777 PO.(H.3F22,0)
6780 WAIT(5)
6785 R(2)=APF.(H.3F22)
```

```
6790 P. #3, F, ' ', #6, Z( 1), V, R 1)
6791 DI
6800 R( 3)=R( 1)-V
6805 R( 4)=R( 2)-W
6850 I F F#2 G. 6865
6855 D( 1)=A BS( R( 3)), D( 2)=V
6856 D( 4)=D( 1)
6860 G. 7000
6865 D( 4)=A BS( R( 3))
6870 I F D( 4)> D( 1) G. 7000
6875 D( 2)=V, D( 1)=D( 4)
7000 I F F=30 G. 7400
7010 I F R( 3)<=1 G. 7014
7012 G. 7020
7014 I F ABS( R( 4))>4 G. 7600
7016 C( 3)=3
7020 I F ABS( R( 3))>2 G. 7600
7023 I F ABS( R( 4))>3 G. 7600
7030 P. 'AFTER  ', #3, F, '  ITERATIONS'
7031 P. V, R( 1), L( 7)
7032 G. 7980
7076 P.
7100 S.
7400 V=D( 2)
7405 G. 4080
7599 ;UPDATING ROUTINE
7600 C( 1)=V+C( 7)* R( 3)/1000
7605 C( 7)=0
7610 I F C( 1)>3000 C( 1)=3000
7900 I F C( 3) #0 G. 7912
7905 V=C( 1)+1
7906 C( 3)=0
7908 G. 7915
7912 V=C( 1)
7915 I F F=N G. 7965
7920 G. 4080
7965 V=C( 1)+1
7968 N=N+15
7970 G. 4080
7980 C( 7)=1
7985 Z( 0)=0
7999 ;DATA TRANSMISSION FROM I-NIC TO LSI 11/23
8000 ;
8010 T=3
8015 P=6
8020 M=P
8025 FI
8030 SC.( C( 6), 8030)
8035 I F M>0 G. 8050
8040 M=P
8045 GOS. 8500
8050 I F Z( 0)>0 G. 8056
8055 G. 8035
8056 DI
8057 I F Z( 1) #32766 G. 8060
8058 S.
8060 P.
8063 P. 'LAMDA1= ', Z( 1), '      LAMDA2= ', Z( 2)
8065 GOS. 8700
8070 Q=Q+1
8072 P. 'U11= ', V, '            U21= ', W
8073 I F ( Z( 1)=M( 0)) & ( Z( 2)=M( 0)) G. 4080
8075 F=0, C( 7)=0
8077 G. 4080
```

```
9080 M=M-1
9085 RTI
9090 S.
9500 Y(0)=7
9505 Y(1)=0
9510 Y(2)=G(7)
9515 Y(3)=X
9520 Y(4)=V
9525 Y(5)=H(7)
9530 Y(6)=I(7)
9535 Y(7)=J(7)
9575 GOS.30000
9580 R.
9700 X=Z(1)
9705 Y=Z(2)
9710 R.
30000 IF Y(0)<=0 G.30020
30010 IF Y(0)<=127 G.30040
30020 P.'S/30000-INVALID WORD COUNT SUPPLIED AS',Y(0),'HENCE STOP'
30030 S.
30040 U=Y(0)*2
30060 GOS.32000
30070 F.J=1 TO Y(0)
30080 U=Y(J)
30090 GOS.32000
30100 U.H.2400(U)
30110 GOS.32000
30120 N.J

30125 U=0
30130 GOS.31000
30140 IF U=0 G.30170
30150 Z(0)=0
30160 R.
30170 U=0
30175 GOS.31000
30180 Z(0)=U/2
30190 IF U#2*Z(0) G.30210
30200 IF U<254 G.30230
30210 P.'S/@30000-INVALID BYTE COUNT RECEIVED AS',U,'HENCE STOP'
30220 S.
30230 F.K=1 TO Z(0)
30235 U=0
30240 GOS.31000
30250 U.H.2400(U)
30260 GOS.31000
30270 U.H.2400(U)
30280 Z(K)=U
30290 N.K
30300 R.
31000 U.H.240A()
31010 U.H.CB(U)
31020 U.H.241A(U)
31030 U.H.240F()
31040 R.
32000 U.H.240A()
32010 U.H.CB(U)
32015 U.H.241A(K)
32020 U.H.240F()
32030 R.
```

```
  50 G.4000
  99 ;TR48 SETTING UP - ATTENUATORS ADJUSTMENT
 100 EI
 101 P=-610,Q=148,R=824,S=1000
 104 G.106
 105 INP.'C11'P,'C12'Q,'C21'R,'C22'S
 106 F.I=1 TO 10
 110 APO.(H.3E00,R)
 120 APO.(H.3E04,S)
 160 PO.(H.3200,0)
 165 PO.(H.3E20,0):WAIT(5)
 170 A=APE.(H.3E20)
 175 PO.(H.3E22,0):WAIT(5)
 180 B=APE.(H.3E22)
 185 P.A,B
 200 WAIT(200)
 250 N.I
 270 G.106
 300 S.
4000 ;PROGRAM 30629
4001 ;LOCAL DECISION UNIT 2
4002 ;INTERACTION BALANCE METHOD WITH LOCAL FEEDBACK
4003 ;
4004 ;
4005 D.(4,4,10,5,5,-1,-1,7,7,7,-1,7,14,14,14,-1,14,14,14,-1,-1,-1,-1,-1
4007 W=1169,X=-2546,Y=-744
4008 G.4010
4010 INP.'U21'W,'LAMDA1'X,'LAMDA2'Y
4011 ;INITIALISATION
4012 Q=1
4015 E=0
4017 N=20
4020 F.I=0 TO 4
4022 A(I)=0,B(I)=0
4023 N.I
4025 F.I=1 TO 5
4027 D(I)=0,E(I)=0,Z(I)=0
4028 N.I
4030 F.I=1 TO 10
4032 C(I)=0
4033 N.I
4035 F.I=1 TO 14
4040 M(I)=0,N(I)=0,O(I)=0,R(I)=0,S(I)=0,Q(I)=0
4045 N.I
4050 P.
4059 P.
4060 P.'INTERACTION BALANCE METHOD WITH LOCAL FEEDBACK'
4062 P.
4064 P.'LOCAL DECISION UNIT 2'
4066 P.
4067 P.'DATE:25-JUL-83'
4068 INP.'TIME'R
4069 P.
4070 P.'U21=1169, LAMDA1=-2546, LAMDA2=-744'
4071 P.
4072 P.
4073 D(1)=32766
4075 P.'  N  LAMDA2  U21  U21*'
4077 ;GAIN COEFFICIENTS FOR CONSTANT GAIN
4078 C(10)=-2500
4079 C(6)=100
4080 Z(0)=0
```

```
4082 E=E+1
6000 ;LOCAL DECISION UNIT '2'
6100 M(1)=1636+3*X/22-4*W/11
6101 N(1)=455+5*W/11-X/22
6102 O(1)=-1045-X/22-W/22
6103 Q(1)=1182+2*X/11+2*W/11
6104 R(1)=1955-X/22-W/22
6105 M(2)=2000+X/6-2*W/3, N(2)=1000
6106 O(2)=-500-W/2
6107 Q(2)=1000+X/6+W/3
6108 R(2)=2500-W/2
6110 M(3)=750-W/2, N(3)=750+W/2
6112 O(3)=-750, Q(3)=0
6113 R(3)=2250
6115 M(4)=1000, N(4)=200+3*W/5-X/10
6117 O(4)=(W-X-13000)/10
6118 Q(4)=800+X/10+2*W/5, R(4)=1700+W/10-X/10
6120 M(5)=1000-W, N(5)=1000
6122 O(5)=-500-W/2, Q(5)=0
6123 R(5)=2500-W/2
6125 M(6)=2000+X/6-2*W/3, N(6)=1000
6127 O(6)=-1000,Q(6)=1000+X/6+W/3
6128 R(6)=3000-W
6130 M(7)=1000, N(7)=1000+W
6132 O(7)=-500+W/2, Q(7)=0
6133 R(7)=2500+W/2
6135 M(8)=1000,N(8)=286+4*W/7-X/14
6137 O(8)=-1000,Q(8)=714+X/14+6*W/14
6138 R(8)=1571+W/7-X/7
6140 M(9)=1000-W, N(9)=1000
6142 O(9)=-1000, Q(9)=0
6143 R(9)=3000-W
6145 M(10)=-1000, N(10)=1000
6147 O(10)=-1000, Q(10)=0
6148 R(10)=3000-W
6150 M(11)=1000, N(11)=1000
6152 O(11)=-1000, Q(11)=W
6153 R(11)=3000-W
6155 M(12)=1000, N(12)=-500+W/2
6157 O(12)=-1000, Q(12)=1500+W/2
6158 R(12)=0
6175 M(13)=1000, N(13)=-1000
6177 O(13)=-1000,Q(13)=2000+W
6178 R(13)=-1000-W
6180 M(14)=1000, N(14)=1000+W
6182 O(14)=-1000, O(14)=0
6183 R(14)=3000+W
6250 F.I=1 TO 14
6255 IF ABS(M(I))>1000 G.6323
6265 IF ABS(N(I))>1000 G.6323
6275 IF ABS(O(I))>1000 G.6323
6285 IF Q(I)<0 G.6323
6290 IF R(I)<0 G.6323
6305 A(1)=((Q(I)-2000)*(Q(I)-2000)/100000)*2
6310 A(2)=(R(I)-3000)*(R(I)-3000)/100000
6315 A(3)=M(I)*M(I)/100000+N(I)*N(I)/100000+O(I)*O(I)/100000
6320 A(4)=Y*W/10000-X*Q(I)/10000
6321 A(0)=A(3)-(105*W/1000)*W/1000-204
6322 G.6325
6323 S(I)=32766
6324 G.6330
6325 S(I)=A(1)+A(2)+A(3)+A(4)+2
6330 N.I
6400 S(0)=32766
```

```
6405 F.I=1 TO 14
6410 IF S(I-1)>S(I) G.6440
6415 S(I)=S(I-1), M(I)=M(I-1), N(I)=N(I-1), O(I)=O(I-1)
6420 Q(I)=Q(I-1), R(I)=R(I-1)
6440 NEXT I
6450 A(1)=((Q(14)-2000)*(Q(14)-2000)/1000)*2
6451 A(2)=(R(14)-3000)*(R(14)-3000)/1000
6452 A(3)=M(14)*M(14)/1000+N(14)*N(14)/1000+O(14)*O(14)/1000
6453 A(4)=Y*W/1000-X*Q(14)/1000
6454 S(14)=A(1)+A(2)+A(3)+A(4)+2
6500 G.6602
6505 P.'C21=',M(14),'     C22=',N(14),'     C23=',O(14)
6510 P.'U21=',W,'     Y21=',Q(14),'     Y22=',R(14)
6516 P.'L2(1000)=',S(14)
6600 ;MODEL & REAL PROCESS INTERFACE
6602 EI
6605 GOS.B500
6607 WAIT(10)
6610 IF Z(0)>0 G.6616
6615 G.6605
6616 IF Z(2)#32766 G.6620
6618 S.
6620 Y=Z(2)
6622 V=Z(3)
6623 H(0)=Z(1), I(0)=Z(2)
6625 D=D+1
6750 APO.(H.3E08,M(14))
6755 APO.(H.3E0A,N(14))
6767 PO.(H.3200,0)
6768 WAIT(500)
6769 PO.(H.3E20,0)
6770 WAIT(5)
6775 R(1)=APE.(H.3E20)
6777 PO.(H.3E22,0)
6780 WAIT(5)
6785 R(2)=APE.(H.3E22)
6790 P.#3,E,' ',#6,Z(2),W,R(2)
6791 DI
6800 R(3)=R(1)-V
6805 R(4)=R(2)-W
6850 IF E#2 G.6865
6855 D(1)=ABS(R(4)),D(3)=W
6856 D(4)=D(1)
6860 G.7000
6865 D(4)=ABS(R(4))
6870 IF D(4)>D(1) G.7000
6875 D(3)=W,D(1)=D(4)
7000 IF E=30 G.7400
7010 IF R(4)<=1 G.7014
7012 G.7020
7014 IF ABS(R(3))>5 G.7600
7016 C(3)=9
7020 IF ABS(R(4))>2 G.7600
7023 IF ABS(R(3))>3 G.7600
7030 P.'AFTER  ',#3,E,'  ITERATIONS'
7031 P.W,R(2),S(14)
7032 G.7980
7100 S.
7400 W=D(3)
7405 G.4080
7599 ;UPDATING ROUTINE
7600 C(2)=W+C(10)*R(4)/1000
7605 C(7)=0
7615 IF C(2)>3000 C(2)=3000
```

```
7820 IF C(2)<=2000 C(2)=2000
7900 IF C(3)#9 G.7912
7905 W=C(2)+1
7906 C(3)=0
7908 G.7915
7912 W=C(2)
7915 IF E=N G.7965
7920 G.4080
7965 W=C(2)+1
7968 N=N+15
7970 G.4080
7980 C(7)=1
7985 Z(0)=0
7999 ;DATA TRANSMISSION FROM I-MIC TO LSI11/23
8000 ;
8010 T=3
8015 P=6
8020 M=P
8025 EI
8030 SC.(C(6),8020)
8035 IF M>0 G.8050
8040 M=P
8045 GOS.8500
8050 IF Z(0)>0 G.8056
8055 G.8035
8056 DI
8057 IF Z(2)#32766 G.8060
8058 S.
8060 P.
8063 P.'LAMDA1=',Z(1),'     LAMDA2=',Z(2)
8065 GOS.8700
8070 O=O+1
8072 P.'U11=',V,'          U21=',W
8073 IF (Z(1)=H(0)) & (Z(2)=N(0)) G.4080
8075 E=0,C(7)=0
8077 G.4080
8080 M=M-T
8085 RTI
8090 S.
8500 Y(0)=9
8505 Y(1)=0
8510 Y(2)=C(7)
8515 Y(3)=Y
8520 Y(4)=W
8525 Y(5)=M(14)
8530 Y(6)=N(14)
8535 Y(7)=O(14)
8540 Y(8)=O(14)
8545 Y(9)=R(14)
8575 GOS.30000
8580 R.
8700 X=Z(1)
8705 Y=Z(2)
8710 R.
30000 IF Y(0)<=0 G.30020
30010 IF Y(0)<=127 G.30040
30020 P.'S/30000-INVALID WORD COUNT SUPPLIED AS',Y(0),'HENCE STOP'
30030 S.
30040 U=Y(0)*2
30060 GOS.32000
30070 F.J=1 TO Y(0)
30080 U=Y(J)
30090 GOS.32000
30100 JUMP-30000(U)
```

```
30110 GOS.32000
30120 N.J
30125 U=0
30130 GOS.31000
30140 IF U=0 G.30170
30150 Z(0)=0
30160 R.
30170 U=0
30175 GOS.31000
30180 Z(0)=U/2
30190 IF U#2*Z(0) G.30210
30200 IF U<254 G.30230
30210 P.'S/@30000-INVALID BYTE COUNT RECEIVED AS',U,'HENCE STOP'
30220 S.
30230 F.K=1 TO Z(0)
30235 U=0
30240 GOS.31000
30250 U.H.2400(U)
30260 GOS.31000
30270 U.H.2400(U)
30280 Z(K)=U
30290 N.K
30300 R.
31000 U.H.2400()
31010 U.H.C8(U)
31020 U.H.2412(U)
31030 U.H.240E()
31040 R.
32000 U.H.2400()
32010 U.H.C8(U)
32015 U.H.2412(K)
32020 U.H.240E()
32030 R.
```